PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	La gestion des groupes de variables en recherche directe
Auteur: Author:	Vincent Garnier
Date:	2010
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Garnier, V. (2010). La gestion des groupes de variables en recherche directe [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/357/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/357/
Directeurs de recherche: Advisors:	Charles Audet
Programme: Program:	Mathématiques appliquées

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

LA GESTION DES GROUPES DE VARIABLES EN RECHERCHE DIRECTE

VINCENT GARNIER
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES ET GÉNIE INDUSTRIEL)
JUIN 2010

© Vincent Garnier, 2010.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

LA GESTION DES GROUPES DE VARIABLES EN RECHERCHE DIRECTE

présenté par : M. GARNIER Vincent
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

M. GAMACHE Michel, Ph.D., président.
M. AUDET Charles, Ph.D., membre et directeur de recherche.
M. DELORME Louis, Ph.D., membre.

À ma famille et mes amis pour leur soutien sans faille,
À mes muses Camille et Pamela pour leur aptitude à colorer ma vie,
À mes collègues Quentin I et II pour leur aide constante et leurs conseils avisés,
À mes labrador et poisson rouge Kitty et Charles Édouard,

Remerciements

Je tiens à remercier beaucoup de personnes, qui m'ont énormément apporté au cours des deux dernières années. Ne souhaitant pas me lancer dans le nominatif hasardeux, je me permets d'user d'une terminologie vague et pseudo-chronologique teintée çà et là d'inserts inopinés.

Un grand merci donc à mes parents, ma soeur, ma famille, mes amies, mes amis, Élie Kakou, la Provence, Vauvenargues, Philippe de Girard, Marie Laurencin, les Arts et Métiers, Boris Vian, le Québec, Polytechnique, Marie-Chantal Toupin, Poly-Théâtre, le Gerad, Avec la lumière le néant, l'Ireq, l'Égotrip, Jean Schulteis, les Cravates, Claude Nougaro, le Polyscope et la C.I.A.

Une mention spéciale à mon directeur de recherche Charles Audet et son acolyte Sébastien Le Digabel, pour leur professionnalisme, leur soutien bienveillant et leur grande disponibilité.

Un prix (spécial également) décerné au vaillant Adrien Fulda, qui m'a occasionnellement nourri et souvent compris, pour son amour de la chanson québecoise, de la grande cuisine et de la politique extérieure de la Tanzanie.

Ma gratitude va en outre à Stéphane Alarie et Louis-Alexandre Leclaire de l'Institut de Recherche d'HYDRO-QUÉBEC, pour m'avoir épaulé tout au long de mon séjour à Varennes, à Michel Gamache de l'École Polytechnique pour avoir accepté gracieusement de présider mon jury, ainsi qu'à mes compagnons de bureau Quentin Lequy, Quentin Reynaud, Majid Salavati et Florian Chambon pour leurs interactions stimulantes au sein de l'univers de la recherche.

Je remercie enfin la machiavélique Camille, pour sa candeur et son professionnalisme.

Résumé

La Recherche par Coordonnées (CS), la Recherche par Motifs Généralisée (GPS) et la Recherche Directe sur Treillis Adaptatif (MADS) sont des exemples d'algorithmes robustes en optimisation non linéaire et non lisse. Afin d'améliorer la solution courante, ces méthodes utilisent des directions d'exploration qui affectent soit une seule variable à la fois (CS), soit plusieurs variables à la fois (GPS et MADS). Nous nous proposons ici de formaliser et généraliser ces approches, à travers le concept de "groupes" de variables : chaque groupe, géré par l'algorithme de manière évolutive, génère des points d'essai en ne modifiant que les variables le concernant. Cela permet la construction d'un voisinage particulier potentiellement fructueux : dans le cas de variables de positionnement par exemple, cela permet de déplacer des objets ou des collections d'objets de manière individuelle. On utilise pour cette étude le logiciel NOMAD développé par Le Digabel (2009), qui est une implémentation écrite en *C++* des tout derniers algorithmes de ce type, à savoir BIMADS et ORTHOMADS, respectivement introduits par Audet *et al*. (2008d) et Abramson *et al*. (2009b). Ces méthodes sont conçues pour l'optimisation de boîtes noires : les évaluations des fonctions relatives aux objectifs et contraintes sont le résultat d'un processus opaque, typiquement un code informatique. Par conséquent, ces fonctions peuvent être non lisses, non linéaires, non convexes ou discontinues, avec possiblement des domaines de définition très fragmentés.

Nous souhaitons également nous atteler à la résolution de problématiques concrètes liées à ce type d'optimisation : nous traiterons en particulier le cas d'un problème de localisation de balises à rayonnement gamma, sur des cartes en deux dimensions à domaines réalisables très fragmentés. Ce projet, mené en collaboration avec l'Institut de recherche d'HYDRO-QUÉBEC (IREQ), vise à améliorer la précision de l'estimation du manteau neigeux et de l'équivalent Eau-Neige, afin de gérer les prévisions hydriques tout au long de l'année, et plus particulièrement aux moments critiques tels que la fonte des neiges printanière.

Abstract

Coordinate Search (CS), Generalized Pattern Search (GPS) and Mesh Adaptive Direct Search (MADS) are examples of robust algorithms for nonsmooth nonlinear optimization. To improve the current solution, these methods use exploratory directions that affect either a single variable at a time (CS), or several variables at once (GPS and MADS). We will formalize and generalize these approaches, through the concept of "groups" of variables : each group, managed by the algorithm dynamically, generates trial points by only changing the variables concerning them. This allows the construction of a particular and potentially fruitful neighborhood : for example, in the case of positioning variables, the algorithm can move objects or collections of objects sequentially. We use for this research the software NOMAD developed by Le Digabel (2009), which is an *C++* implementation of the very latest MADS algorithms, namely BIMADS and ORTHOMADS, respectively introduced by Audet *et al*. (2008d) and Abramson *et al*. (2009b). These methods are designed for blackbox optimization : the evaluations of the objective and constraint functions are the result of an opaque process, typically a computer code. Therefore, these functions may be nonsmooth, non-linear, non-convex or discontinuous, with possibly highly fragmented domains.

We aim to solve practical issues linked to this type of optimization : we will focus on the case of a gamma-monitoring beacons location problem, on two-dimensional maps with very fragmented domains. This project, in collaboration with the Research Institute of HYDRO-QUEBEC (IREQ), aims to improve the snowpack estimate accuracy in order to manage the hydrological forecast throughout the year, especially at critical times such as spring snowmelt.

Table des matières

Liste des tableaux

Liste des figures

Liste des algorithmes

Sigles, abréviations et notations

Sigles et abréviations	
BIMADS	MADS pour les problèmes biobjectif.
CS	Coordinate Search.
DFO	Derivative-Free Optimization.
DIRECT	DIviding RECTangles.
GMON	Gamma-MONitoring devices.
GPS	Generalized Pattern Search.
IREQ	Institut de Recherche d'HYDRO-QUÉBEC.
ISATIS	Logiciel interne à l'IREQ.
LTMADS	Lower Triangular MADS.
MADS	Mesh Adaptive Direct Search.
NOMAD	Nonlinear Optimization with MADS.
ORTHOMADS	MADS déterministe avec directions orthogonales.
PSDMADS	Parallel Space Decomposition with MADS.
VNS	Variable Neighborhood Search.

Notations	
$\mathbb{N}$	Ensemble des entiers naturels positifs.
$\mathbb{Z}$	Ensemble des entiers relatifs.
$\mathbb{Q}$	Ensemble des rationnels.
$\mathbb{R}$	Ensemble des réels.
$\mathbb{R}^+$	Ensemble des réels positifs.
Ω	Domaine réalisable.
f_Ω	Fonction objectif avec barrière sur le domaine Ω.
x	Candidat de $\mathbb{R}^n$.
$\|x\|$	Norme euclidienne de x.
$B_\epsilon(x)$	Boule ouverte de rayon ϵ centrée en x.

N	Ensemble des indices des variables composant le vecteur x.
N_q	Sous-ensemble ordonné (ou *groupe*) d'indices de variables inclus dans N.
$\overline{N_q}$	Ensemble des indices de N n'appartenant pas au groupe N_q.
$\Omega_q(x)$	Sous-ensemble de Ω propre au groupe N_q au point x.
$F_q(x)$	Sous-espace affine propre au groupe N_q passant par x.
k	Compteur d'itérations.
x_k	Centre de sonde à l'itération k.
y	Point d'essai à tester à l'itération k.
$\bar{y}$	Point d'essai réalisable le plus proche de y.
M_k	Treillis à l'itération k.
Δ_k^m	Paramètre de taille du treillis à l'itération k.
Δ_k^p	Paramètre de taille de sonde à l'itération k.
D	Ensemble de directions engendrant positivement $\mathbb{R}^n$.
D_k	Ensemble des directions de sonde à l'itération k.
P_k	Voisinage de sonde à l'itération k.
S_k	Ensemble des points à tester au cours de l'étape de recherche à l'itération k.

Dénominations algorithmiques	
	Groupement initial
A	Abscisses et ordonnées (2 groupes de $n/2$ variables).
I	Variable individuelle (n groupes d'une seule variable).
P	Paires de coordonnées alternées ($n/2$ groupes de 2 variables).
T	Toutes ensemble (un seul groupe de n variables).
X	Paires par GMON ($n/2$ groupes de 2 variables).
	Algorithme de reconfiguration
D	Dynamique avec fusion par distance.
F	Dynamique avec fusion par mouvement.
K	Dynamique avec classification par *K-means*.
M	Dynamique avec gestion par mouvement.
S	Statique.
V	Dynamique avec régression linéaire.
W	Dynamique avec régression linéaire et transformation des rangs.
	Règle secondaire
IR	Variable individuelle.
TR	Toutes ensemble.
XR	Paires par GMON.
	Autres
BET	Budget d'évaluations total.
BPG	Budget d'évaluations par GMON.
EC	Critère d'arrêt basé sur les échecs consécutifs.
G	Éxécution terminant par un algorithme MADS classique.
N	Éxécution avec groupe N ajouté à chaque itération.

Chapitre 1

INTRODUCTION

L'art de l'optimisation d'un système réside dans la détermination, en termes de paramètres, du compromis idéal, au vu de contraintes et de critères établis. En ingénierie, bien souvent, le problème est complexe : les fonctions utilisées pour le modéliser ne vérifient pas d'hypothèses de linéarité, de dérivabilité ou même de continuité, et même si les dérivées existent, celles-ci sont difficiles à approximer avec précision. En fait, tel qu'indiqué par Stephens et Baritompa (1998), il n'existe, sauf cas particuliers [1], aucun algorithme assurant la convergence vers un optimum global. Et quand bien même on parviendrait à atteindre ce dernier, prouver qu'il est effectivement global est en général un défi non trivial et consommateur en ressources.

De fait, il est nécessaire de diviser notre examen en deux étapes étroitement liées : *la recherche locale*, qui consiste à chercher, à partir d'une solution réalisable, un optimum local aux alentours de celle-ci, et *la recherche globale*, qui consiste essentiellement à effectuer un bilan des résultats des recherches locales et à en relancer sur diverses parties du domaine réalisable, de façon à ne négliger aucun des espaces de solutions potentiellement fructueux. Parmi les classes de méthodes efficaces n'utilisant pas de dérivées, on retrouve notamment la recherche directe sur treillis adaptatifs. Celle-ci, nommée MADS pour "Mesh Adaptive Direct Search", est conçue pour résoudre les problèmes d'optimisation non lisse difficiles dits en *boîtes noires*. Ces algorithmes visitent le voisinage d'une solution candidate (avec l'espoir de l'améliorer) grâce à des directions d'exploration évolutives. Ces dernières, jusqu'à présent, reposent sur des critères indépendants du problème traité.

Cependant, certains problèmes comportent une structure telle que certaines variables sont liées entre elles *a priori*. Ceci arrive par exemple lorsqu'un sous-ensemble des variables

[1] Dans le cas d'un programme convexe, par exemple, tout optimum local est global, ce qui facilite les choses.

représente les coordonnées d'un objet réel. Ce genre de cas est fréquent, puisqu'il englobe au moins tous les problèmes de localisation. S'il est évident que le modélisateur sait à quels paramètres correspondent chaque objet, on n'exploite cependant pas ou peu le lien qui unit ces variables descriptives. Il peut être intéressant, comme nous le verrons, de modifier certains objets ou groupes d'objets séquentiellement.

Toutefois, il arrive très souvent qu'on ne possède aucune information *a priori* sur la structure du problème, et on ne connaît donc pas à l'avance quelles sont les variables possiblement liées. Il existe des méthodes d'analyse de sensibilité destinée à déterminer s'il existe des groupes de variables ayant certaines affinités ou corrélations.

Nous avons pour objectif de déterminer des méthodes de recherche directe qui "ajustent" le voisinage exploré en fonction non seulement du problème traité mais également du déroulement de l'algorithme. La solution que nous proposons est de ne modifier qu'un sous-ensemble de variables à la fois, en adaptant les directions de recherche. La principale tâche consiste à choisir de façon pertinente les sous-ensembles ou *groupes* de variables à déplacer individuellement.

Nous allons nous intéresser, dans ce mémoire, à la détermination et à l'exploitation des liens existants entre les groupes de variables, notamment dans le cas où ces dernières ont une structure de positionnement. En bref, il s'agit de l'étude de la gestion des groupes de variables dans le cadre de la recherche directe. Nous allons tout d'abord commencer par formuler le problème et les hypothèses, puis nous entamerons un tour d'horizon des algorithmes conçus pour l'optimisation de *boîtes noires*.

1.1 L'optimisation de boîtes noires

On considère ici l'optimisation de boîtes noires dans un contexte général. Dans cette approche, l'optimiseur se place extérieurement au problème, comme s'il "enfermait" ce dernier dans une boîte noire. Ceci est utile par exemple lorsque le fonctionnement interne du système est soit inaccessible, soit délibérément omis. Le caractère inaccessible est fréquent en ingénierie, par exemple pour des raisons physiques (codes informatiques opaques, processus complexes) ou commerciales (secret technologique, propriété industrielle). On s'emploie essentiellement à varier les entrées et analyser les sorties de façon à identifier le

plus rapidement possible les paramètres d'entrée qui produisent les meilleures sorties.

Tout problème d'optimisation ou *programme mathématique* peut se ramener à la forme suivante :

$$\min_{x \in \Omega} \quad f(x)$$

où $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$, $f, c_j : X \to \mathbb{R} \cup \{\infty\}$ pour tout $j \in J = \{1, 2, \ldots, m\}$, et X est un sous-ensemble de $\mathbb{R}^n$.

La minimisation n'est pas restrictive, dans la mesure où minimiser la fonction objectif f revient à maximiser $g = -f$. Dans notre contexte, les évaluations de f et des fonctions de contraintes c_j sont habituellement le résultat d'un code informatique opaque, dit *boîte noire*. Le domaine de définition de f, noté Ω, est défini au travers de l'ensemble X et des fonctions c_j, ce qui nous permet de distinguer :

1. Les contraintes *non relaxables* définissent X et ne peuvent être violées par aucun point d'essai. Par exemple, une valeur booléenne indique si la simulation peut s'exécuter.
2. Les contraintes *relaxables* $c_j(x) \leq 0$ peuvent être violées, et la simulation sera exécutée, mais $c_j(x)$ fournit une mesure de la violation.
3. Les contraintes *cachées* sont un terme pratique pour définir l'ensemble des points dans la région réalisable pour les contraintes non relaxables à laquelle la boîte noire échoue à renvoyer une valeur pour l'une des fonctions du problème. Un exemple typique est lorsque la simulation échoue inopinément.

Ces contraintes, comme nous allons le voir au chapitre 4, peuvent être gérées de manière spécifique. Enfin, pour des raisons explicitées ci-après, les dérivées des fonctions f et c_j sont indisponibles et difficiles à approximer avec une précision suffisante.

1.1.1 Hypothèses

Pour caractériser ces boîtes noires, on se place dans le cas général du problème de conception issu de l'ingénierie.

Premièrement, les fonctions définissant le problème sont en général le produit de codes informatiques, qui peuvent renvoyer des valeurs binaires (tests booléens), contenir des conditions (**if** , **else** , **or** , ...) ou encore des sauts (**goto**). De plus, l'acquisition et/ou le calcul des valeurs des fonctions peuvent introduire un certain bruit et des perturbations, qui empêchent l'utilisation de dérivées ou d'approximations de dérivées. Troisièmement, dans les cas concrets où le modèle n'est pas abusivement simplifié, les fonctions sont en général coûteuses à évaluer, et les réponses peuvent requérir de quelques secondes à plusieurs jours d'attente. Finalement, les évaluations des fonctions peuvent échouer inopinément même pour $x \in \Omega$ et seules quelques valeurs correctes sont assurées.

Ces considérations nous mènent aux hypothèses suivantes :

1. f, c_j peuvent être non lisses, non linéaires, non convexes et discontinues.
2. Le domaine X peut être irrégulier, non convexe et fragmenté.
3. L'approximation précise des dérivées est problématique.
4. L'évaluation des fonctions peut échouer même pour $x \in \Omega$.
5. Les fonctions sont coûteuses à évaluer.

Tout d'abord, nous allons introduire les principaux concepts employés dans la suite du mémoire. Il s'agit ici de donner au lecteur un avant-goût du principe de la recherche par motifs.

1.1.2 Concepts-clés en Recherche Directe

Les algorithmes de recherche directe par motifs s'articulent à notre sens autour de quatre notions fondamentales et intimement liées qu'il convient d'introduire : les *bases positives*, le *treillis*, l'*étape de sonde*, et l'*étape de recherche*. De plus, dans le contexte général de l'optimisation de boîtes noires, il nous semble pertinent d'ajouter à la liste précédente le très utile concept de *fonction substitut*.

Base positive Une base positive est une famille de vecteurs engendrant l'espace $\mathbb{R}^n$ par des combinaisons linéaires positives, telle qu'aucun sous-ensemble propre n'a cette propriété. Ces bases servent à générer les directions d'exploration du point courant. Il est

légitime de se poser la question : pourquoi introduire un tel concept alors que la notion de base est univoque en algèbre linéaire ? La réponse est un peu technique : leur importance vient du fait qu'elles contiennent au moins un élément dans tout demi-espace ouvert. De fait, les algorithmes d'optimisation qui les utilisent sont assurés de posséder toujours au moins une direction de descente parmi leurs possibilités (voir Figure 1.1), ce qui permet de réaliser une analyse de convergence théorique hiérarchique dépendant du degré de régularité de la fonction. Rappelons enfin que l'exploitation de cette notion en analyse de convergence a été faite a posteriori, et qu'il ne s'agit donc pas d'un compromis entre théorie et performance pratique, mais bien d'une justification pertinente de l'efficacité de tels algorithmes sur les problèmes d'ingénierie réels.

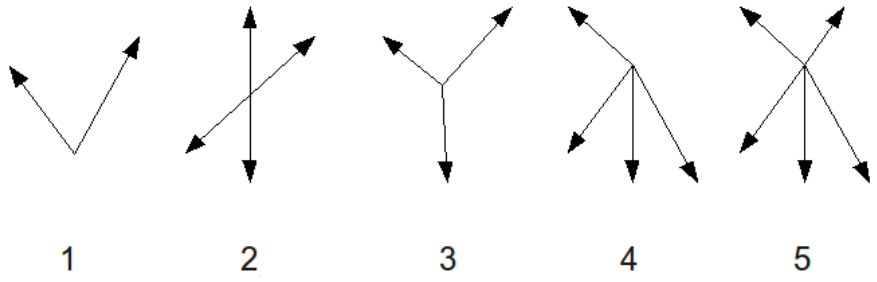

FIGURE 1.1 Exemples bidimensionnels de bases positives : 1 est une base mais non une base positive, tandis que 2 et 3 le sont. Enfin, 4 et 5 ne le sont pas, car l'un n'engendre pas positivement $\mathbb{R}^2$, et l'autre n'est pas minimal.

Treillis Le treillis est une grille de points quadrillant un espace $\mathbb{R}^n$. Il s'agit d'un sous-ensemble discret de $\mathbb{R}^n$, dont la finesse (i.e. la distance entre deux points) à l'itération k dépend du *paramètre de taille du treillis*. Pour assurer la cohérence de l'analyse de convergence, tous les points évalués doivent impérativement appartenir au treillis. Celui-ci se ramifie lorsque l'algorithme échoue à trouver des points meilleurs que le candidat actuel, et croît ou stagne dans le cas contraire. Ainsi, en considérant un critère d'arrêt basé sur la taille du treillis, l'algorithme converge (sous certaines conditions que nous préciserons

ultérieurement) vers un point satisfaisant des conditions d'optimalité de premier ordre.

Étape de sonde (Poll step) C'est de cette étape, reposant sur des règles strictes et aboutissant en cas d'échec à une diminution de la taille du treillis, que découlent les propriétés de convergence hiérarchiques. Soit une base positive de $\mathbb{R}^n$ prédéfinie. À chaque itération, nous construisons avec les vecteurs de cette base, au prix éventuellement de transformations plus ou moins complexes, une nouvelle famille constituant les directions de sonde. Le voisinage de sonde est alors composé de points sur le treillis avoisinant l'itéré courant dans les directions de sonde. C'est-à-dire que nous générons un nombre fini de points d'essai autour de notre candidat courant avec pour objectif d'en trouver un diminuant la valeur de la fonction objectif. En cas de succès, cette solution remplace le candidat courant et la taille du treillis est augmentée ou conservée. Dans le cas contraire, on raffine le treillis, et le candidat solution actuel est dit "optimimum local du treillis".

Bref, la base positive définit la *direction* de sonde et les paramètres du treillis définissent la *norme* du vecteur, de façon à former les points candidats voisins qui vont être testés lors de l'étape de sonde.

Étape de recherche (Search step) Dans l'étape de recherche, la fonction objectif est évaluée en un nombre fini de points sur le treillis courant, avec l'objectif d'obtenir un candidat améliorant la meilleure valeur objectif courante. Si un tel point est trouvé, il peut immédiatement remplacer le centre de sonde actuel. Cette étape, libre de règles pourvu qu'elle soit d'une durée finie, est généralement d'une grande importance au sein du processus d'optimisation.

En effet, si l'étape de sonde garantit en théorie des propriétés de convergence fortes, il est important, en pratique, d'effectuer une étape de recherche pertinente et efficace, afin de garantir d'une part une convergence rapide et d'autre part, une possible évasion des optima locaux dans laquelle la sonde pourrait être piégée du fait de sa portée réduite.

Ainsi, lorsque le détenteur du problème d'optimisation connaît *a priori* une stratégie permettant de guider l'algorithme vers les régions fructueuses (e.g., où les minima locaux sont de bonne qualité), il est intéressant de l'inclure au sein de MADS.

Fonction substitut Les problèmes que nous désirons traiter comprennent des fonctions coûteuses à évaluer. Or, la convergence des algorithmes de recherche directe est généralement lente, du fait précisément de l'indisponibilité des dérivées (qui empêche toute approximation locale par développement de Taylor). Nous pouvons alors introduire l'emploi de *fonctions subsituts*, servant à choisir soigneusement les candidats à évaluer à moindre coût, avec l'espoir de diminuer significativement le nombre d'évaluations nécessaire à l'obtention d'une bonne solution.

La fonction substitut n'a pas besoin d'être une excellente approximation de la fonction originale. Elle doit simplement posséder une structure relativement semblable, dès lors que son utilisation se base sur l'hypothèse implicite qu'un optimum local pour le surrogate doit être également une solution de qualité pour le problème original. On décline deux sortes de substituts, à savoir les modèles *non adaptatifs* (non recalibrés) et les modèles *adaptatifs* (recalibrés en fonction des valeurs des fonctions originales obtenues au fur et à mesure du déroulement de l'algorithme), qui possèdent chacun leurs propres avantages. Les modèles fixes, non évolutifs, sont généralement plus simples à mettre en place lorsqu'on dispose d'information sur la boîte noire et requièrent moins de temps de calcul. L'approche adaptative, plus sophistiquée, permet de corriger les substituts de mauvaise qualité, ce qui peut parfois être salutaire.

Nous allons voir, dans ce mémoire, que l'emploi de fonctions substitut est souvent nécessaire (par exemple losque la boîte originale est inaccessible), et qu'elle peut permettre l'obtention d'une solution de qualité tout en réalisant une économie substantielle en termes d'évaluations.

1.2 Objectifs

Nous cherchons à couvrir les problèmes avec groupes de variables, incluant ceux de localisation, qui ont les caractéristiques suivantes :

1. Dimension relativement faible ($n \leq 100$).
2. Évaluations des fonctions via des boîtes noires (voir hypothèses à la section 1.1.1).
3. Sous-domaines de définition pour les groupes :

- $2D$: de “non fragmenté” à “très fragmenté”.
- $3D$ et supérieur : “non fragmenté”.

Pour situer le lecteur parmi les acronymes, nous utilisons dans notre cas un algorithme de la classe MADS (*Mesh Adpative Direct Search* pour *Recherche Directe sur Treillis Adaptatif*), qui est une extension de GPS (*Generalized Pattern Search* pour *Recherche par Motifs Généralisée*), qui est lui-même une généralisation de CS (*Coordinate Search* pour *Recherche par Coordonnées*). MADS possède des propriétés de convergence hiérarchique fortes, reposant sur le calcul non lisse de Clarke (1983), qui seront explicitées à la section 4.2.3.

1.2.1 Motivations

La version de l’algorithme MADS que nous développons a pour objectif de résoudre des problèmes d’optimisation d’une forme en adéquation avec les hypothèses précédentes. La flexibilité de cette approche “boîte noire” nous permet l’adoption d’une démarche générale de résolution, fonctionnant aussi bien sur des domaines fragmentés que réguliers. Plus précisément, nous traitons dans ce mémoire le cas général des groupes de variables (à la section 4) et nous étudions un problème de localisation particulier, puisque les variables de position sont incluses dans le cadre plus général des groupes de variables. Dans beaucoup de cas concrets où les fonctions d’évaluation sont coûteuses, les méthodes MADS sont extrêmement efficaces lorsqu’utilisées avec des fonctions substitut. Bien sûr, cela requiert une fonction substitut appropriée modélisant le comportement de la boîte noire originale, sans quoi nous pourrions introduire un biais indésirable.

Les problèmes de localisation

Il existe de nombreux cas où les variables représentent les positions d’objets concrets dans l’espace à deux ou trois dimensions. Renouer avec ce caractère concret en ne déplaçant non pas des ensembles de variables de façon aléatoire, mais des objets de façon séquentielle et intelligente, permet en général un meilleur résultat. Dans l’exemple de la localisation

des GMON pour l'estimation du manteau neigeux (décrit à la section 3), l'utilisation de fonctions substitut nous permet à la fois d'accélérer la recherche d'optima et d'éviter les nombreux calculs de modules de krigeage, qui sont coûteux en temps et en ressources humaines.

Les problèmes avec variables prédominantes

Dans de nombreux problèmes, certaines variables (ou certains groupes de variables) ont une influence prépondérante sur le système. Si ce dernier est paramétré par un grand nombre de facteurs, l'identification des variables principales permet la gestion d'une pré-optimisation salutaire en termes d'évaluations. Cela repose sur une analyse de sensibilité au fur et à mesure du déroulement de l'algorithme, qui oriente ses recherches de façon à accélérer la convergence. Souvent, le propriétaire de la boîte noire possède une idée générale, si ce n'est même parfois précise, des facteurs influents, ce qui, dans notre cas, facilite la démarche de l'optimiseur et appuie l'intérêt d'un tel travail.

En s'appuyant sur le travail qui a déjà été accompli dans le domaine, on souhaite prolonger et renforcer l'étude des algorithmes de la classe MADS dans le cas des groupements de variables, tout en s'assurant que les propriétés fondamentales de convergence hiérarchique sont conservées. Nous précisons ici les objectifs, qui nous permettront de réaliser une revue de la littérature pertinente en identifiant les problèmes similaires à ceux étudiés.

1.2.2 Plan

L'accomplissement de l'objectif principal passe par la réalisation de plusieurs étapes. Tout d'abord, nous allons effectuer une revue de la littérature pour les problèmes de localisation de type boîte noire, i.e., en optimisation non lisse et non linéaire, avec une présentation informelle de MADS, ORTHOMADS (la toute dernière instanciation de MADS), ainsi que NOMAD (l'implémentation logicielle utilisée). Ensuite, nous décrirons de manière préliminaire le problème de localisation de l'IREQ qui a motivé ce travail afin d'en saisir la logique. À la section 4, nous formaliserons et généraliserons le problème théoriquement, en présentant notamment plus en détail l'algorithme MADS ainsi que l'extension de son

analyse de convergence, puis nous verrons quelles sont les modifications à apporter pour conserver les propriétés de convergence hiérarchiques.

Finalement, nous en viendrons à la pratique à la section 5. Une implémentation algorithmique et des tests empiriques menés sur une “boîte noire” substitut à celle de l'IREQ, servira alors à obtenir d'une part un ensemble de résultats numériques préliminaires constituant un premier banc d'essai, et d'autres part des directions de travail pour la création d'outils, de fonctions et de paramétrisations adaptés à la gestion des groupes de variables. On fera ensuite une synthèse pour les directions fructueuses qui permettront ensuite à NOMAD d'être plus efficace sur le le problème concret de positionnement des balises gamma pour l'IREQ à la section 3, afin d'obtenir un excellent compromis entre qualité de solution (au vu des critères qui seront définis) et nombre d'évaluations de boîte noire.

Une discussion finale sur les résultats obtenus et les pistes à développer clôturera le mémoire.

Chapitre 2

REVUE DE LITTÉRATURE

Nous cherchons ici à identifier, de façon non exhaustive, des problèmes similaires traités dans des travaux antérieurs, afin de dresser un bilan des directions et idées testées jusqu'ici, de saisir les fondements de nos motivations et d'exhiber les (nombreuses) applications possibles de ce travail. Nous allons tout d'abord présenter quelques algorithmes d'optimisation sans dérivées, en insistant un peu plus particulièrement sur ceux qui nous concernent (MADS et son instanciation ORTHOMADS), en mentionnant notamment les principaux résultats d'analyse de convergence. Ensuite, nous nous concentrerons sur les travaux et méthodes traitant des groupes de variables.

2.1 Les algorithmes de recherche directe pour boîte noire

Une importante caractéristique commune aux algorithmes de recherche directe réside dans leur capacité à traiter uniquement avec les valeurs numériques des fonctions, sans avoir recours au calcul ou même à l'estimation des dérivées. Ces algorithmes, à partir d'une solution initiale, se déplacent itérativement vers une (meilleure) solution voisine, grâce à différentes directions d'exploration, éventuellement variables et basées sur des transformations géométriques. Bien sûr, cela sous-entend que la notion de voisinage a un sens pour le problème considéré, et qu'on est capable de définir explicitement au moins une solution voisine pour un candidat donné.

2.1.1 Rétrospective en recherche directe

On pourra noter tout d'abord, à titre d'exemples de premières méthodes efficaces, la méthode évolutionnaire de Box (1957), l'algorithme original de recherche par motifs de

Hooke et Jeeves (1961), la méthode de Spendley *et al*. (1962) (inspirée de Box) introduisant l'utilisation de simplexes en recherche directe, la méthode basée sur les directions conjuguées de Powell (1964) et enfin le fameux algorithme de Nelder et Mead (1965) basé sur des transformations simpliciales. GPS, développé par Torczon (1997), MADS introduit par Audet et Dennis, J. E., Jr. (2006a), DIRECT issu de Jones *et al*. (1993) et DFO créé par Conn *et al*. (1998) sont des exemples d'algorithmes récents.

Ces méthodes ne fonctionnent en général qu'en faible dimension (bien que certains développements récents, comme l'algorithme PSDMADS de Audet *et al*. (2008b), permettent de gérer jusqu'à plusieurs centaines de variables), et partagent sensiblement les mêmes avantages et inconvénients. Ils sont communément reconnus pour leur robustesse, leur (relative) simplicité de mise en oeuvre, ainsi que leur efficacité sur un grand nombre de problèmes "de terrain" en ingénierie. Toutefois, dès lors qu'on traite avec des structures connues exploitables, telles que celles linéaires, convexes, différentiables ou lisses, de telles méthodes sont (sauf exceptions) lentes et inapropriées. Il sera alors évidemment préférable d'utiliser des algorithmes plus spécialisées, comme les méthodes du *Simplexe* (programmation linéaire), de *Branch-and-Cut* (linéaire en nombres entiers), de *Newton* ou encore celle de *Broyden-Fletcher-Goldfarb-Shanno* dite *BFGS* (non linéaire différentiable). Enfin, à l'origine basés sur la pratique, les algorithmes de recherche directe ne disposaient pas, avant 2002, d'analyse de convergence compatible avec les problèmes traités. Cette lacune tend depuis à être comblée grâce (entre autres) au développement de la théorie du calcul non lisse. Citons quelques exemples de méthodes notables.

L'agorithme DFO est basé sur une méthode de région de confiance. Le lecteur intéressé pourra se reporter à sa description aux articles Conn *et al*. (1997a), Conn *et al*. (1997b) et pourra trouver quelques exemples d'applications pratiques dans l'article Conn *et al*. (1998).

L'algorithme DIRECT est une méthode d'échantillonnage déterministe pour l'optimisation avec contraintes de bornes, basée sur la division géométrique de l'espace en rectangles (généralisés à des hyper-rectangles en dimension n), renforcée par une analyse de convergence dans Finkel et Kelley (2004).

L'algorithme Nelder-Mead progresse grâce à des opérations simpliciales telles que les réflexions, en se déplaçant dans la direction opposée au point ayant la pire valeur. Il trouve de nombreux exemples d'applications en ingénierie et microtechnologie, tel qu'indiqué

dans Dennis, J. E., Jr. et Woods (1987). Bien qu'il ait été montré des faiblesses en matière de convergence dans McKinnon (1998), des preuves pour le cas en faible dimension, telles que celle de Lagarias *et al*. (1998) et des variantes renforcées dans divers domaines de la conception d'ingénierie ont vu le jour, parmi lesquelles Kelley (1999), Price *et al*. (2002), Nazareth et Tseng (2002) ou encore Luersen (2004).

Les algorithmes de type MADS, qui sont une généralisation de GPS, utilisent quant à eux la notion de base positive, ainsi que celles de treillis adaptatifs quadrillant l'espace de recherche.

Nous pouvons en fait catégoriser ces différentes approches issues de la recherche directe en deux familles. La première regroupe les méthodes tentant de bâtir un modèle de la fonction à optimiser, ce qui est souvent effectué par approximation polynômiale à partir de points obtenus grâce à un échantillonnage. On retrouve dans cette catégorie l'algorithme DFO, ainsi que les algorithmes qui tentent d'approximer le gradient de f pour déterminer les zones prometteuses à explorer.

La seconde famille contient les algorithmes qui ne tentent pas directement d'approximer de gradients ou de construire de modèles de f, qui comptent (entre autres) les méthodes basées sur la géométrie du simplexe (Nelder-Mead et variantes), celles reposant sur un échantillonnage séquentiel de l'espace (DIRECT et variantes), et finalement les méthodes de recherche par motifs généralisée (GPS et variantes). Nous allons nous intéresser par la suite à cette dernière catégorie, dont les algorithmes MADS, présentés à la section 4.2, font partie.

Notons pour finir qu'il existe un grand nombre d'applications pratiques destinées aux algorithmes de recherche directe, tel que mentionné dans Conn *et al*. (2009), en physique, ingénierie (conception mécanique, optimisation des procédés), chimie (géométrie moléculaire, pollution de l'air), électronique (nanotechnologie, conception de circuit), ou encore en économie (tarification dynamique). Tel que montré dans Audet et Orban (2006), il est même possible d'optimiser les paramètres algorithmiques propres aux algorithmes destinés à résoudre un problème d'optimisaton !

2.1.2 De GPS à MADS

Les algorithmes GPS ont été définis par Torczon (1997), puis renforcés par une analyse dans Lewis et Torczon (1999) et Lewis et Torczon (2000) respectivement pour l'optimisation sous contraintes de bornes et sous contraintes linéaires. À l'origine, ces algorithmes puisaient leurs directions de recherche dans des ensembles prédéfinis, et possédaient à ce titre un nombre restreint de déplacements possibles, ce qui constituait un inconvénient majeur au niveau de l'analyse de convergence théorique. Ce cadre a été plus tard unifiée par Audet et Dennis, J. E., Jr. (2003), ce qui a donné naissance aux algorithmes de la classe MADS. Sous ce paradigme, les algorithmes peuvent désormais générer (possiblement aléatoirement) des combinaisons entières de directions, élargissant le champ des possibilités. Tous ces algorithmes sont caractérisés par une série de déplacements exploratoires autour du point courant, formant des motifs (*patterns*), selon les directions d'une base positive. À chaque itération, la fonction objectif est évaluée sur un sous-ensemble des points du motif. Si une amélioration est trouvée, le point associé est accepté comme nouveau point courant, et la taille du prochain motif est conservée ou augmentée. Sinon, un nouveau motif, de taille réduite, est généré autour de l'ancien point courant.

On pourra notamment citer divers exemples de renforts et d'adaptation des méthodes *pattern search* pour l'optimisation sans dérivées : couplages avec du recuit simulé par Hedar et Fukushima (2004), avec des algorithmes de sélection stochastiques par Sriver *et al*. (2004), contrôle de la précision par Polak et Wetter (2006), techniques d'exploitation selon la structure des problèmes par Price et Toint (2004), ou encore leur mise en parallèle par Hough *et al*. (2001).

Pour éviter de noyer le lecteur sous un flot de notations, nous reportons la description technique de l'algorithme MADS ainsi que son analyse de convergence à la section 4.2.3. Ainsi, l'analyse du cas classique sera directement confrontée à celle dans le cas des groupes de variables. Nous allons néanmoins présenter superficiellement la toute dernière implémentation des algorithmes de cette classe par Abramson *et al*. (2009b), ORTHOMADS, ainsi que le logiciel NOMAD que nous avons employé tout au long de notre travail.

2.1.3 Une instanciation particulière : ORTHOMADS

Contexte de naissance Débutons tout d'abord par la première instanciation de MADS, nommée LTMADS. Avec celle-ci, les directions de sonde sont générées de façon non déterministe par l'intermédiaire d'une matrice triangulaire inférieure aléatoire. Ainsi, LTMADS possède à la limite un nombre infini de directions de sonde et l'union sur toutes les directions normalisées et sur l'ensemble des itérations devient dense dans la sphère unité avec une probabilité égale à un. En contrepartie cependant, on peut observer (cf. Custódio et Vicente (2007)), à une itération donnée, des angles importants entre les directions de sonde de LTMADS, indésirables car laissant de larges zones insondées et donc ralentissant la convergence. C'est afin de pallier ces lacunes qu'une nouvelle version a été développée par Abramson *et al*. (2009b) : ORTHOMADS.

Avantages La différence entre cette version et LTMADS repose sur leur façon de générer les directions de sonde. Avec ORTHOMADS, ces dernières sont générées quasi-aléatoirement et orthogonalement les unes aux autres. Il y a deux avantages à cela. Premièrement, leur caractère orthogonal leur permet d'être bien réparties dans l'espace et de ne pas laisser inexplorées de larges portions de domaine. Deuxièmement, leur caractère quasi-aléatoire permet des exécutions déterministes et donc reproductibles sur n'importe quelles machines de calcul.

Exemple 2.1.1 *Avec* ORTHOMADS*, à chaque itération k, une direction est choisie selon une suite quasi-aléatoire de Halton (1960), à partir de laquelle une base positive orthogonale normée formée de $2n$ directions est construite. On obtient ainsi un ensemble de candidats voisins P_k tels que les $p^i, i \in \{1, ..., 2n\}$ sur la Figure 2.1. Dans celle-ci, nous avons modifié chacune des composantes du point x_k pour générer les points candidats voisins. Par la suite, et c'est l'objet de ce mémoire, nous aimerions générer des directions ne déplaçant que certaines variables choisies, afin de structurer davantage le déroulement de l'algorithme.*

Finalement, il a été montré dans Abramson *et al*. (2009b) que les performances numériques d'ORTHOMADS sont en général très bonnes pour l'optimisation lisse et non

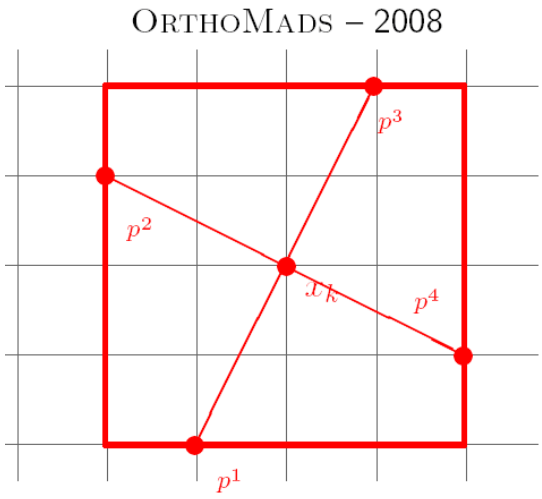

FIGURE 2.1 Exemple de directions de sonde en deux dimensions ($n = 2$) avec ORTHOMADS. Issu de la présentation *Mesh Adaptive Direct Searches for Constrained Optimization, Dennis, 2008*.

lisse, que les problèmes soient contraints ou non. À présent, présentons le logiciel NOMAD, dont l'implémentation est basée sur les précédents algorithmes.

2.2 Le logiciel NOMAD

NOMAD est une implémentation *C++* des algorithmes MADS, développée par Abramson *et al*. (2003) et codée par Le Digabel (2009), conçue pour l'optimisation sous contraintes de boîtes noires de la forme décrite dans l'introduction 1.1. Pour éviter une certaine redondance dans la présentation des travaux sur MADS, j'introduis ici des extensions liées à MADS tout en listant quelques fonctionnalités notables de NOMAD.

Variables de catégorie L'analyse de MADS a été étendue par Abramson *et al*. (2009a) au cas des variables de catégorie [1], montrant que l'algorithme conserve ses propriétés de convergence. NOMAD, depuis la version 3.3 sortie en novembre 2009, gère ce type de variables de manière spécifique.

Variables périodiques NOMAD gère les variables périodiques en projetant les points d'essai hors de la période dans l'intervalle défini grâce à une opération de modulo. Il est montré dans Audet et Le Digabel (2009) que les conditions algorithmiques de convergence sont respectées et que cette méthode hérite directement de l'analyse de convergence de MADS.

Optimisation biobjectif NOMAD utilise dans ce cas l'algorithme BIMADS décrit dans Audet *et al*. (2008d), capable de résoudre les problèmes de la forme :

$$\min_{x \in \Omega} F(x) = (f_1(x), f_2(x))$$

L'algorithme lance une suite d'exécutions de MADS sur des reformulations mono-objectif du problème. Le front de Pareto, composé de la liste des points non dominés, est progressivement approché grâce aux évaluations réalisées au cours de ces exécutions.

Couplage avec VNS Dans Audet *et al*. (2008a), les auteurs proposent de coupler l'algorithme MADS avec un algorithme de type VNS (en étape de recherche), pour offrir un certain équilibre entre recherche locale et diversification globale. Le principe consiste à créer un ou des nouveaux points de départ en perturbant la solution courante avec une certaine amplitude et à relancer l'algorithme de descente locale (MADS) à partir de ces points. Cette perturbation permet (éventuellement) d'échapper aux minima locaux. En effet, si la descente échoue à améliorer la solution courante, l'amplitude est augmentée, et ce jusqu'à un certain seuil qui constitue le critère d'arrêt. Dans le cas contraire, l'amplitude est réinitialisée à son plus faible niveau, et le nouveau minimum atteint devient la solution courante.

[1] Les variables sont dites *catégoriques* ou *qualitatives nominales*, lorsqu'elles ne peuvent être hiérarchisées ou classées selon une gradation logique.

Groupes de variables Des groupes de variables peuvent être définis. Par la suite, à chaque étape de sonde de l'algorithme MADS, des directions différentes seront générées pour chaque groupe (ces directions ne modifiant que les variables appartenant au groupe). Chaque groupe sera ensuite déplacé séquentiellement selon ses directions. Par exemple, pour un problème de localisation, si on partitionne l'ensemble des variables de façon à ce que chaque groupe représente un objet spatial $3D$ unique, ceux-ci seront déplacés sur leur domaine de définition $3D$ respectif de manière individuelle (un par un). L'étude des groupes de variables constitue l'objet de ce mémoire.

Par ailleurs, les paramètres constants à caractère (relativement) arbitraire, tels que les conditions d'arrêt, la taille du treillis initiale, les points de départs, la graine de Halton, le type d'algorithme MADS ou le type des directions, sont manipulables par l'utilisateur. Il est de la responsabilité de ce dernier de considérer l'influence de ces facteurs, mais, étant donnés le nombre de paramètres possiblement modifiables et le type de problème traité (optimisation de boîte noire à évaluation coûteuse), il serait illogique de s'y attarder avec une "vraie" boîte noire. On peut alors utiliser une fonction substitut pour faire ces choix. Un cadre rigoureux a d'ailleurs été établi pour l'utilisation de fonctions substituts par Booker *et al*. (1999). Il faut donc dégager les principaux paramètres susceptibles d'avoir une influence significative sur la qualité du résultat (aptitude de MADS à déterminer l'optimum global) et la convergence algorithmique (nombre d'évaluations de boîte noire et temps nécessaire pour obtenir cet optimum).

2.3 Analyse des groupes de variables

Qu'il s'agisse de classifier, de repérer des motifs communs, de former des agrégats, de mettre à jour des anomalies ou des corrélations, l'analyse des variables ou des groupes de variables offre des possibilités riches. Elle est d'ailleurs déjà utilisée dans de nombreux domaines, comme en traitement d'images, en exploration de données, en économétrie, en sondage, et plus généralement dès lors qu'on se livre à des statistiques sur des échantillons de population. Cependant, ici, le cadre d'utilisation est tout à fait original. Nous nous

servons des groupes de variables pour “aider” un algorithme de recherche locale à générer des directions de sonde fructueuses. Notre problème n’a pas de structure a priori : aucune hypothèse d’indépendance, de normalité des valeurs ou de linéarité. Il s’agit simplement d’utiliser les données (valeurs des variables et objectif), afin de bâtir un algorithme qui va gérer des groupes de variables. L’analyse de sensibilité offre à cet égard des outils intéressants qu’il convient de mentionner.

2.3.1 Analyse de sensibilité

On souhaite étudier les conséquences des perturbations des paramètres d’entrée sur la valeur de la fonction objectif. On peut considérer les entrées comme des variables aléatoires indépendantes. On pourra citer les nombreuses contributions de Sobol et Saltelli sur le sujet, notamment Sobol (1993), Saltelli et Scott (1997), Saltelli *et al*. (2000), qui ont introduit des méthodes de décomposition de la variance, ainsi que le livre de Saporta (2006), constituant une excellente introduction à l’application de méthodes probabilistes et statistiques à l’analyse de données, et la thèse de Jacques (2005) qui référence clairement un grand nombre d’importants travaux. Enfin, le livre de Saltelli *et al*. (2004) consitue un tutorial concret intéressant.

On peut définir deux types de méthodes d’analyse de sensibilité, celles dites *locales*, et celles dites *globales*, qui, comme leur nom l’indique, tentent de déterminer localement (e.g., dans le voisinage d’un point donné) ou globalement (sur l’ensemble du domaine) les relations entre facteurs. Parmi celles locales, on peut noter l’utilisation du Gradient Simplexe, qui a été introduite dans les méthodes de recherche directe par Custódio et Vicente (2007), puis a été approfondie par Custódio *et al*. (2008) pour lui fournir un support et une justification théorique dans le cas non contraint. Parmi celles globales qui nous semblent adaptées au contexte présent, on peut citer la régression multiple, l’utilisation de ratios de corrélation standardisés, le partitionnement de données (*clustering*), ainsi que les méthodes de décomposition de variance telle que ANOVA *(ANalysis Of VAriance)* ou FAST *(Fourier Amplitude Sensitivity Test)*. On peut alors dériver des indices de sensibilité standards, éventuellement multidimensionnels, associés aux variables et aux groupes de variables. Toutefois, la décomposition de la variance et la plupart des méthodes statistiques

nécessitent soit des échantillons de taille significative (par exemple dans le cas de méthodes non paramétriques), soit des hypothèses sur le modèle que nous ne pouvons être certains de vérifier dès lors qu'une boîte noire possède, par définition, une structure inconnue *a priori*.

2.3.2 Les problèmes propices aux groupes

Voici les deux catégories de problèmes qui semblent (au premier abord) intéressants de traiter par l'approche de groupement de variables : les cas où certains groupes de variables jouent des rôles relativement *symétriques* ; ceux où, au contraire, les rôles sont fortement *assymétriques*. La première catégorie contient entre autres la classe des problèmes de localisation, tandis que la seconde englobe tous les problèmes illustrant par exemple la loi de Pareto, selon laquelle une minorité de paramètres est à l'origine d'une majorité d'effets.

Rappelons tout d'abord que le groupement des variables dans le cas de MADS a déjà été abordé par Audet *et al*. (2008b) avec la conception de l'algorithme parallèle PSDMADS. Dans cet article, il est question de réaliser un algorithme MADS parallèle efficace pour les problèmes de boîte noire avec un grand nombre de variables, de façon à palier à la difficulté de MADS à traiter des problèmes de plus d'une centaine de variables. L'ensemble des variables est partitionné aléatoirement en groupes de taille assez proche par un processus maître, qui attribue ces groupes à des processus esclaves. Ainsi, chacun de ces derniers travaille en modifiant seulement le sous-ensemble de variables qui lui a été assigné. PSDMADS fonctionne de manière asynchrone, et c'est au processus maître d'assurer la liaison entre les processus esclaves et la cohérence de l'algorithme en fonction des résultats obtenus. Un processus tournant en parallèle sur le problème complet permet d'assurer la convergence sur le plan théorique.

Les problèmes de localisation Par le passé, notre équipe a travaillé à deux reprises avec des variables de positionnement. Dans Fowler *et al*. (2008), nous avons étudié la question du placement de pompes afin d'extraire un contaminant répandu dans une nappe sous-terraine. Dans Audet *et al*. (2005), nous avons étudié le déploiement de bouées de détection de tsunamis dans le Pacifique afin de minimiser le temps d'avertissement aux villes côtières menacées, et il fut notamment évoqué dans la discussion le déplacement individuel de

bouées.

Par ailleurs, dans Alberto *et al*. (2004), un algorithme GPS est employé pour des problèmes de géométrie moléculaire, où l'étape de recherche repose sur de faibles perturbations aléatoires sur l'ensemble des positions, avec pour but de déterminer la configuration des atomes de façon à minimiser l'énergie totale. Des comparaisons de méthodes d'optimisation sans dérivées ont été effectuées dans Fowler *et al*. (2008), où le problème résidait au niveau de l'approvisionnement de nappe souterraine relatives à des problèmes de collectivité en hydrologie.

Notons également l'atelier préliminaire Audet *et al*. (2008c) qui introduit le problème de localisation de balises GMON, que nous traiterons dans ce travail. Dans ce projet concret de localisation, la recherche locale sera assurée par MADS. Cependant, afin de s'évader d'un optimum local, il faut habituellement recourir à une métaheuristique (intégrée dans l'étape de recherche), dont le travail consiste essentiellement à relancer des recherches locales à travers le domaine de solutions afin de comparer des optima locaux.

Les problèmes avec variables prépondérantes Il existe en fait un grand nombre de problèmes où les variables ne jouent pas un rôle symétrique, à l'inverse des problèmes de localisation où on tente (généralement) au contraire d'obtenir une configuration équilibrée. Il s'agit alors d'identifier les variables ou les groupes de variables qui ont une influence notable.

Par exemple, au cours d'un projet aéronautique mené en collaboration avec *Boeing*, l'équipe de Booker *et al*. (1998) a travaillé sur un problème d'optimisation en boîte noire comportant 31 variables. Après une analyse de sensibilité, il a été révélé que seulement 11 variables parmi celles en jeu étaient prépondérantes. Ainsi, en se concentrant sur ces dernières en priorité (par exemple en fixant les autres variables), il a été possible de résoudre le problème en économisant un temps de calcul significatif.

Chapitre 3

INTRODUCTION À LA LOCALISATION DE BALISES GMON POUR LA MESURE DE L'ÉQUIVALENT EAU-NEIGE

Nous allons présenter dès ce chapitre le problème de l'IREQ (Institut de recherche d'HYDRO-QUÉBEC), afin de permettre au lecteur de mieux cerner nos motivations. Le travail sur les groupes de variables a très nettement été articulé autour de cette application, c'est pourquoi nous avons cru bon de l'introduire avant d'en venir à des aspects plus techniques. Décrivons tout d'abord précisément le contexte et les enjeux de notre travail.

3.1 Contexte énergético-économique

Selon le Ministère des Ressources Naturelles et de la Faune (2004), l'hydroélectricité représente 97% de l'ensemble de l'énergie produite au Québec, où la société d'État HYDRO-QUÉBEC (en situation de quasi-monopole) est responsable de la production, du transport et de la distribution de l'électricité. L'unité qui nous concerne, relative à la production, se nomme *"Prévision et Qualité des données Hydroélectriques"*. Elle est chargée de produire les prévisions d'apports d'eau pour plus de 90 bassins au Québec, qui s'étendent sur un vaste territoire d'environ $550\,000\ km^2$.

Ces prévisions influencent significativement l'efficacité de la gestion des centrales hydro-électriques, ce qui n'est pas négligeable dès lors que l'eau représente 95% de la puissance de production d'HYDRO-QUÉBEC. De plus, la production, si gérée adéquatement,

peut générer un surplus d'électricité qu'HYDRO-QUÉBEC peut ensuite vendre aux provinces environnantes et aux États-Unis. Rappelons que, selon Audet *et al*. (2008c), les transactions générées par la vente de seulement 5% de son électricité aux États-Unis représentent 25% des profits de la compagnie.

3.1.1 L'estimation du manteau nival

Par ailleurs, la neige représente environ 30% de la variation des débits sur l'ensemble du territoire, et cette valeur valeur atteint 40% si l'on se limite aux régions nordiques. On entrevoit ici l'importance d'effectuer une estimation précise du manteau nival. De plus, selon Alarie et Garnier (2009), durant la fonte printanière, dans les réservoirs les plus larges, la plupart de l'eau requise pour la production annuelle d'énergie est reçue en deux à trois semaines. Si un réservoir n'est pas dimensionné de façon apropriée, il peut y avoir de graves inondations, pouvant entraîner des dommages environnementaux et des gaspillages énergétiques colossaux, engendrant ultimement des pertes économiques substantielles (cf. Environnement Canada (2009) pour des données chiffrées). Finalement, la capacité des centrales au cours de l'eau peut être excédée si celles-ci ne sont pas gérées convenablement, entraînant pertes et risques supplémentaires.

Pour éviter tout problème, nous devons répondre à deux questions indépendantes : *"où la neige fond-elle ?"* et *"quelle est la quantité de neige qui fond ?"*. L'estimation de l'équivalent eau-neige concerne cette dernière interrogation et nous fournit une indication précieuse sur les flux d'eau résultant de la fonte des neiges.

3.1.2 Les réseaux de mesure actuels

Pour parvenir aux prévisions d'apports d'eau, HYDRO-QUÉBEC dispose de plusieurs méthodes de collecte (voir Fig. 3.1). Cependant, le *réseau des observations hydrométéorologiques* est actuellement lacunaire et le demeurera vraisemblablement, en raison de la très grande taille du territoire à couvrir et du coût élevé de l'installation d'appareils. En outre, l'actuel *réseau des mesures in situ* est basé sur le prélèvement manuel et ponctuel de carottes de glace, ce qui ne permet pas d'effectuer des mesures de manière complète et régulière sur l'ensemble du territoire. Enfin, combiner ces réseaux à des méthodes géostatistiques ne

suffit pas à atteindre une précision satisfaisante (HYDRO-QUÉBEC souhaite obtenir une erreur inférieure à 20%).

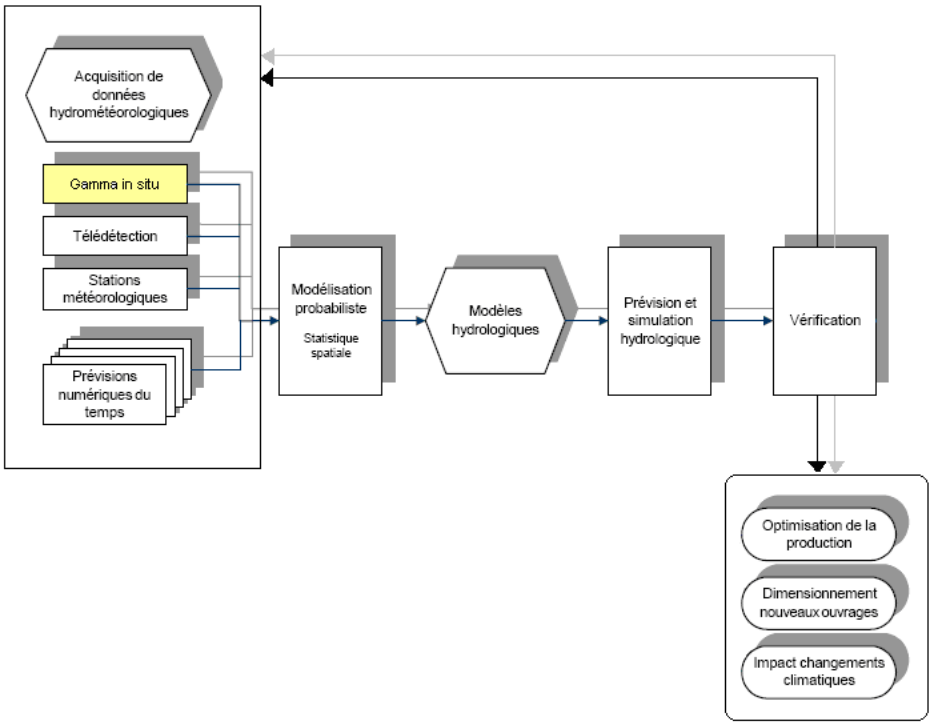

FIGURE 3.1 Chaîne de gestion des prévisions hydriques au sein d'HYDRO-QUÉBEC.

Pour corriger cela, d'autres initiatives ont été testées ou sont actuellement déployées, notamment basées sur la *télédétection*, le *rayonnement cosmique* ou encore les *émissions gamma*. Si, à l'heure actuelle, aucune de ces méthodes (prises individuellement) n'est à la fois satisfaisante et capable de couvrir l'ensemble du territoire (pour des raisons pratiques, technologiques ou financières), la combinaison de plusieurs technologies pourraient permettre de surmonter ces difficultés.

3.1.3 Les GMON

La mesure du rayonnement *gamma in situ* est effectuée grâce au déploiement de balises, nommées GMON pour *"Gamma MONitoring"* (Fig. 3.2), sur le territoire québecois. Plus précisément, l'intensité du rayonnement gamma terrestre perçu par le panneau recepteur de la balise, atténuée par la présence d'eau entre le sol et la balise, permet d'en mesurer la quantité (tout état confondu) localement.

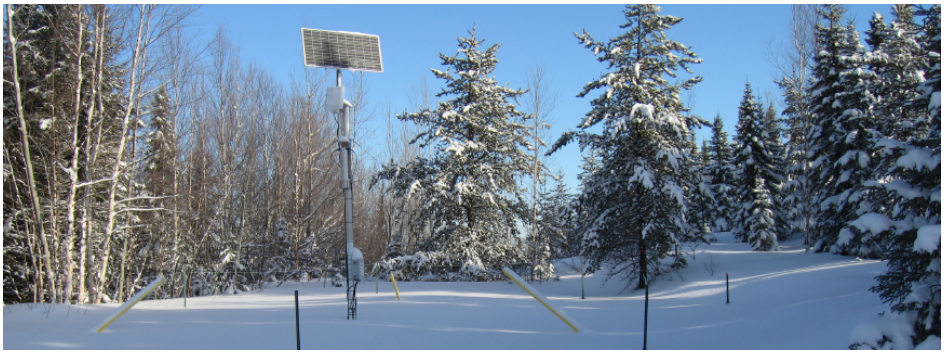

FIGURE 3.2 Une balise GMON destinée à affiner l'estimation de l'équivalent eau-neige. Issu de Alarie et Garnier (2009).

La *télédétection*, quant à elle, permet de déterminer entre autres l'état (liquide, solide) de l'eau. Les informations recueillies peuvent alors être recoupées et complétées par les stations météorologiques et les sonars déjà en place. Ces derniers permettent entre autres de mesurer l'épaisseur du manteau nival. Enfin, une méthode de *krigeage avec dérive externe*, proposée par Tapsoba *et al*. (2005), permet l'interpolation des mesures locales sur l'ensemble du territoire.

Ainsi, l'estimation globale de l'équivalent eau-neige passe par la connaissance locale de la hauteur, l'état et la quantité d'eau du manteau nival : il est alors possible de déduire la densité, ce qui permet d'avoir une idée précise du phénomène de fonte (lieu, vitesse, quantité), et plus particulièrement de prévoir la *crue*, partie majeure de la fonte se déroulant sur seulement une dizaine de jour. Une amélioration de la précision conduirait à de meilleures connaissances de l'apport hydrique à venir en période de fonte, à un suivi efficace du stock

d'énergie, ainsi qu'à une gestion plus fine des ouvrages hydroélectriques, ce qui serait à terme bénéfique aux niveaux sécuritaire et économique.

Or, la localisation des balises GMON, dont les coûts de déplacements ne sont pas négligeables, influence de manière prépondérante la précision de l'estimation de l'équivalent eau-neige, dont la qualité des prévisions dépend largement : les positions des balises GMON sont donc à choisir avec soin.

3.2 Le problème

Le problème qui nous concerne réside dans le positionnement optimal de balises GMON sur les cartes de certains bassins gérés par HYDRO-QUÉBEC, avec pour objectif de minimiser l'erreur sur l'estimation de l'*équivalent eau-neige*, paramètre essentiel à l'approximation du manteau neigeux. Le nombre de balises à placer, bien qu'inconnu, n'est pas directement considéré comme une variable du système pour des raisons évoquées à la section 3.2.2.

3.2.1 Le domaine

HYDRO-QUÉBEC désire positionner les balises GMON sur certains bassins appartenant au territoire québecois. Cependant, pour des raisons physiques, celles-ci ne peuvent être posées que sur une fraction du domaine total considéré. Rappelons que HYDRO-QUÉBEC gère quelques 565 barrages, 75 réservoirs et 56 centrales hydro-électriques, situés sur 90 bassins. Le territoire (et par là même le domaine de définition du problème de localisation) est de fait immense. Les GMON ne peuvent donc être disposés qu'en certains sites précis. Plusieurs banques de données sont disponibles pour chaque bassin, que nous utiliserons pour définir à la fois le domaine *réalisable* (où il est possible de poser un GMON), et celui *estimé* (où l'on estime par krigreage le manteau neigeux, d'après entre autres les mesures des GMON). Cependant, ces régions sont fortement fragmentées et discontinues. Il s'agit d'un problème qui est, de par sa nature, propice au groupement de variables. À titre d'exemple, la Figure 3.3 représente les domaines estimé et réalisable du bassin de "Saint-Maurice". Les pixels noirs représentent les points où on peut positionner des GMON.

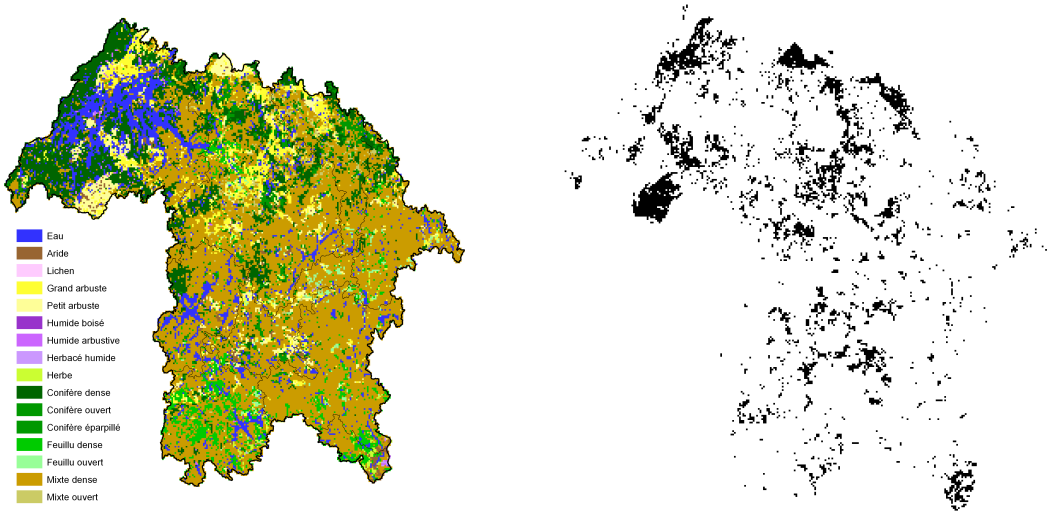

FIGURE 3.3 La carte du bassin de Saint-Maurice (STM) avec une résolution de $1\ km$: à gauche le domaine total avec les différentes composantes physiques, à droite le domaine réalisable (en noir) uniquement.

Les problèmes de positionnement ne peuvent être traitées indépendamment pour chaque carte de bassin que si les interactions entre les GMON d'un bassin et la fonction objectif d'un autre bassin sont jugées négligeables. Cela pose la question de l'échelle de résolution du problème et des éventuels regroupements de cartes.

L'échelle du problème

Nous pouvons considérer uniquement un bassin à la fois, en découpant le déploiement global en sous-problèmes indépendants, ou encore regrouper des ensembles de bassins. Il semble certain que l'idéal serait de considérer le problème à la plus grande échelle possible, pour entrevoir les interactions entre les GMON des différents bassins, et qu'il pourrait être néfaste pour les solutions de ne se mettre qu'à l'échelle d'un bassin uniquement. En effet, imaginons deux bassins ayant une frontière commune (ou suffisamment proches), alors une balise de GMON posée proche de la frontière d'un bassin sera considéré comme une mauvaise solution si jamais nous nous concentrons sur un bassin uniquement. Cependant,

en considérant les deux bassins en même temps, une balise frontalière pourrait couvrir les deux bassins et permettre d'obtenir une bonne solution.

On pourrait donc soit se placer à la plus grande échelle possible, soit, au vu de la portée des GMON (leur zone d'influence sur les mesures), déterminer si certains groupes de bassins doivent être traités ensemble ou non. Ainsi, nous regrouperons les cartes des bassins proches et nous pourrons traiter séparément les bassins isolés.

Toutefois, si la gestion par les clients de l'IREQ se fait par bassin, une inter-dépendance pourrait poser des difficultés d'ordre pratique.

3.2.2 Modélisation préliminaire

Les hypothèses

Après concertation avec les experts de l'IREQ, nous supposons que le nombre de GMON par carte varie de 5 à 10. Ces derniers peuvent être placés partout sur leur carte de région réalisable (définie par certaines composantes physiques). Les positions de certains GMON sont fixées *a priori*, notamment pour assurer la continuité des mesures issues du réseau *in situ*.

La valeur de la résolution des cartes est finalement fixée à $1\ km$, afin que les résultats soient exploitables. On se laisse la possibilité de regrouper les cartes des bassins. Le problème à optimiser, conçu à l'interne par l'IREQ, n'est pas accessible de l'extérieur. De plus, le processus menant au calcul des valeurs de fonction objectif est le résultat d'un code informatique complexe et opaque basé sur l'utilisation de méthodes de krigeage et la génération de variogrammes : ce processus peut donc être considéré de notre point de vue comme une "boîte noire". Il n'est pas du ressort de l'optimiseur de concevoir cette dernière, et nous laisserons l'IREQ décider de la fonction objectif : la dépendance entre la portée d'un GMON et son lieu d'implantation et le coût d'implantation des GMON (éventuellement fonction du lieu) y seront directement intégrés. Notons que, sans prise en compte des coûts, l'erreur de mesure est une fonction monotone décroissante du nombre de GMON : ce dernier ne peut alors constituer une variable du problème d'optimisation, sans quoi l'algorithme préconiserait systématiquement l'emploi du plus grand nombre de GMON possibles.

Étant donnée la faible plage de variation du nombre des GMON ($n \in \{5, 6, \ldots, 10\}$), une

stratégie commode consiste à déterminer pour chaque configuration potentielle donnée (n GMON, dont ℓ fixes) le positionnement optimal. Nous pourrons alors laisser décider HYDRO-QUÉBEC, qui connait le coût de l'implantation des GMON, à partir de quel nombre de GMON cela est rentable. Toutefois, il est tout à fait envisageable de prendre en compte ce coût (qui devra alors être modélisé par HYDRO-QUÉBEC) pour réaliser le travail d'optimisation avec un nombre variable de GMON.

Formulation des objectifs

Nous souhaitons à l'origine minimiser l'erreur relative à l'estimation de l'équivalent eau-neige. Bien sûr, l'incorporation de pénalités propres aux coûts de déploiement est envisagée, mais pour rester simple, énonçons clairement l'objectif du travail :

Objectif primaire *Étant donnés n* GMON*, dont ℓ sont fixes, quelle est la position optimale des $n - \ell$* GMON *libres qui minimise l'erreur sur les mesures ?*

3.2.3 Plan préliminaire

Ce problème de positionnement de balises sur des cartes connues offre une structure propice au groupement de variables, notion que nous avons jusqu'ici survolée. Au prochain chapitre, nous allons définir et développer le cadre général des groupes de variables ainsi qu'une discussion sur le caractère potentiellement fragmenté du domaine. Au chapitre suivant, nous allons bâtir un modèle *substitut* pour se livrer à des tests préliminaires, avec l'objectif de déterminer une stratégie de groupement satisfaisante en vue de la résolution du problème original que nous venons de décrire. Ainsi, le prochain chapitre abordera les groupes de variables de façon conceptuelle, le suivant explicitera la constitution d'un banc d'essais, et le dernier traitera de la résolution du problème de l'IREQ.

Chapitre 4

LES GROUPES DE VARIABLES

Bien que MADS soit à l'origine une méthode conçue pour résoudre les problèmes d'optimisation dépourvus de structure particulière, nous allons nous attaquer ici à la détermination et à l'exploitation des liens existants entre les variables, notamment dans le cas où ces dernières ont une structure de positionnement [1]. Pour ce faire, nous allons considérer une stratégie additionnelle qui pourrait avoir une influence bénéfique sur la convergence algorithmique : le *groupement de variables*.

Au cours de l'algorithme MADS, lors de la constitution de l'ensemble de recherche globale ou de sonde, se pose le problème du choix du nombre k de variables parmi les n constituant le vecteur $x \in \mathbb{R}^n$ qui vont varier simultanément, i.e., du nombre de variables affectées par chaque direction à chaque itération de NOMAD, avant d'évaluer la fonction objectif. Faut-il que ce soit toutes les variables ou seulement un sous-ensemble d'entre elles ?

Précédemment, NOMAD modifiait toutes les variables simultanément, mais il est possible, depuis la version *3.1* (sortie en février 2009), de varier seulement des sous-ensembles de ces variables pour générer nos points d'essai. À l'origine, nous étions motivés par le fait que, dans les problèmes de localisation par exemple, les variables indiquent des coordonnées pour un seul et même objet dans l'espace. Modifier un groupe de variables représentant un objet ou un ensemble d'objets aurait donc un sens plus concret pour ce type de problème que varier simultanément les n variables ensemble (MADS classique) ou séparément (Recherche par Coordonnées). Cette méthode serait susceptible de guider l'algorithme non seulement vers une meilleure solution mais surtout en un nombre réduit d'évaluations. De plus, dans les cas où la fragmentation du domaine Ω est importante, il

[1] I.e., lorsque toutes ou une partie des variables correspondent aux coordonnées d'une collection d'objets à positionner.

peut être difficile de définir des configurations voisines proches réalisables en déplaçant toutes les variables en même temps. Il est alors judicieux d'opter pour une approche moins agressive, en ne changeant qu'un sous-ensemble des variables à la fois.

Dans le contexte de cette étude, nous avons raffiné une approche développée précédemment en permettant à l'algorithme de partitionner les variables en groupes, et de varier les variables d'un seul groupe à la fois. Cette structure de groupe a ensuite été intégrée de façon générique dans NOMAD. On peut observer sur la Figure 4.1 les déplacements typiques de trois balises selon les stratégies GPS, MADS et la nouvelle. GPS utilise un seul paramètre du treillis et ne déplace les balises que selon les directions d'une base positive prédéfinie (ici une seule variable varie), tandis que MADS en possède deux et déplace toutes les balises à la fois, disposant d'un plus large éventail de possibilités. La nouvelle stratégie permet quant à elle de déplacer les balises individuellement, tout en disposant pour chaque mouvement d'une grande variété d'alternatives.

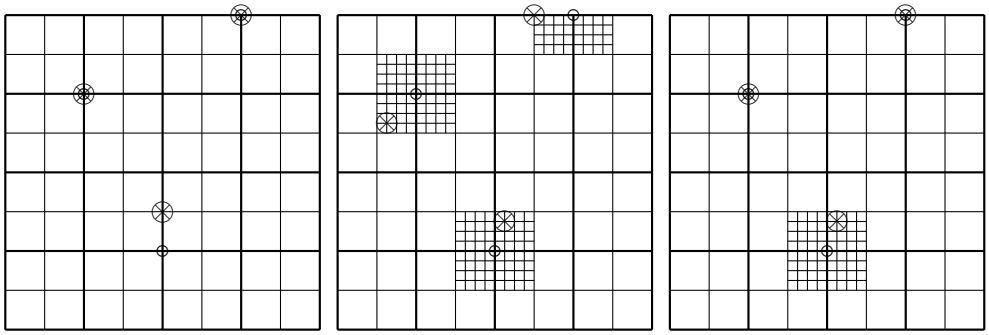

FIGURE 4.1 Les petits cercles représentent le positionnement initial de trois balises, et les autres signes $\otimes$ représentent le positionnement obtenu par une sonde locale de NOMAD. Pour la nouvelle stratégie, dans cet exemple, chaque balise constitue un groupe de deux variables. De cette manière, une seule balise est déplacée à la fois.

Nous conservons ici uniquement les hypothèses admises à la section 1.1.1, même si les stratégies pratiques proposées supposent implicitement la structure du problème adaptée aux regroupements de variables (e.g., avec variables de positionnement ou variables prépondérantes). Les premières observations numériques sur des boîtes noires artificielles (voir section 5.4) tendent à montrer que la gestion des groupes de variables comportent

des aspects prometteurs, tout en conservant les résultats de l'analyse de convergence (section 4.3).

4.1 Formulation

Nous allons dans un premier temps proposer une formulation que nous pensons adaptée à la gestion des groupes de variables, afin de mettre en place une terminologie commune et rigoureuse.

L'extension de MADS au cas des groupes est en fait une généralisation du cas classique. Nous choisissons ici de générer les points de sonde en ne modifiant qu'un sous-ensemble de variables donné (ou *groupe*) à la fois. Nous listons alors tous les points de sonde obtenus pour chacun des groupes, construisant P_k, afin de procéder aux évaluations. Ainsi, en cas de succès, nous ne pouvons modifier (ou déplacer) qu'un groupe de variables à la fois.

4.1.1 Notations pour les groupes

Pour des raisons de clarté, on se place ici à itération fixe (sauf mention contraire), ce qui nous évite l'ajout d'indice k associé au numéro des itération. Notons que grâce à la barrière progressive (définie plus loin à la section 4.2.1), il est possible également de traiter les candidats x hors de Ω. On introduit les notations suivantes :

- Soit un candidat $x \in \mathbb{R}^n$.
- Soit N l'ensemble des entiers $\{1, 2, \ldots, n\}$.
- Soit $N_q \subseteq N$, $q \in Q$ un sous-ensemble ordonné d'indices de variables, que l'on appellera *groupe de variables* ou plus simplement *groupe*.
- Soit $n_q = |N_q|$ la cardinalité du groupe N_q.
- Soit $\overline{N_q} = N \backslash N_q$ l'ensemble des indices de N n'appartenant pas à N_q.

Par abus de langage, nous emploierons par la suite les expressions commodes "*une ou des variables forment un groupe*" ou "*un groupe contient une ou des variables*" pour signifier respectivement "*les indices relatifs à une ou des variables forment un groupe*" ou "*un groupe contient les indices relatifs à une ou des variables*".

Dans le cas général, nous disposons de $|Q|$ groupes de variables, dont l'union constitue la totalité des indices des variables, i.e. $\bigcup_{q \in Q} N_q = N$. Notons également que l'intersection entre les groupes peut être non nulle, c'est-à-dire qu'une variable peut tout à fait appartenir à plusieurs groupes différents.

Remarque 4.1.1 *Dans l'instanciation classique de* MADS *(avant la version 3.1 de* NOMAD*), toutes les variables sont modifiées en même temps. C'est-à-dire que l'ensemble des variables forment un seul groupe $N_q = N$, donc $n_q = n$ et $|Q| = 1$. De plus, dans le cas particulier d'*ORTHOMADS*, il y a $2n$ points de sonde à tester.*

4.2 L'algorithme MADS

Nous allons dans un premier temps rappeler rigoureusement le déroulement de l'algorithme MADS ainsi que son analyse de convergence dans le cas classique. Les notions importantes (base positive, treillis, étape de sonde, étape de recherche) ont déjà été évoquées dans l'introduction, nous ne reviendrons donc dessus que brièvement. La gestion des contraintes n'ayant pas encore été abordée, nous allons la présenter ici.

4.2.1 Gestion des contraintes

On se propose de définir les trois stratégies propres à la gestion des contraintes pour MADS, à savoir la barrière extrême (EB), la barrière progressive (PB) et la barrière progressive-à-extrême (PEB).

La barrière extrême (EB) Dans l'instanciation originale de MADS issue de Audet et Dennis, J. E., Jr. (2006b), toutes les contraintes sont traitées par barrière extrême. Cette approche peut être interprétée de cette manière : nous traitons le problème sans contraintes en remplaçant la fonction objectif f par f_Ω, définie ainsi :

$$f_\Omega(x) := \begin{cases} f(x) & \text{si } x \in \Omega, \\ \infty & \text{sinon.} \end{cases}$$

Cette approche ne nous permet pas de partir d'un point qui ne satisfait pas les contraintes non relaxables. Cependant, comme nous le verrons, il est possible d'utiliser cette méthode en deux phases.

La barrière progressive (PB) MADS-PB, issu de Audet et Dennis, J. E., Jr. (2009), traite les contraintes relaxables $c_j(x) \leq 0$ différemment des non relaxables définissant X. On utilise ici une fonction de violation de contraintes non négative $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$:

$$h(x) := \begin{cases} \sum_{j \in J} \left(\max\left(c_j(x), 0\right)\right)^2 & \text{si } x \in X, \\ \infty & \text{sinon.} \end{cases}$$

Les fonctions de violation de contrainte dans ce contexte servent à l'application de méthodes de filtre, introduites originellement par Fletcher *et al*. (1998), et étendue au cas de l'optimisation non lisse dans Audet et Dennis, J. E., Jr. (2004). Une revue peut-être trouvée dans Fletcher *et al*. (2006). Une propriété de la fonction de violation de contrainte est que $x \in \Omega$ si et seulement si $h(x) = 0$. De plus, si $0 < h(x) < \infty$ alors x satisfait les contraintes non relaxables mais pas toutes les relaxables : $x \in X \backslash \Omega$. Un seuil $h_k^{\max} \geq 0$ est imposé sur la valeur de violation de contrainte. La valeur de ce seuil suit une simple fonction décroissante avec le nombre d'itérations k. Le seuil est utilisé pour rejeter les points d'essai $x \in \mathbb{R}^n$ générés à l'itération k tel que $h(x) > h_k^{\max}$. On appelle cela l'approche avec *barrière progressive*.

La barrière progressive-à-extrême (PEB) Dans le cas où aucun des points initiaux n'est irréalisable, mais que l'on souhaite utiliser la barrière extrême en seconde phase, nous introduisons une troisième stratégie pour gérer les contraintes relaxables : la *barrière progressive-à-extrême* (PEB) consiste à traiter initialement les contraintes par barrière progressive. Dans ce cas, si le sondage autour du point irréalisable génère un nouveau point irréalisable qui satisfait une contrainte violée par le centre de sonde, alors cette contrainte sera à l'avenir traitée non plus comme une barrière progressive mais comme une barrière extrême. De même, le lecteur souhaitant plus de précisions, notamment au niveau de l'analyse de convergence, est invité à se référer aux articles Audet *et al*. (2010),

Abramson *et al*. (2009b) et Audet et Dennis, J. E., Jr. (2009).

4.2.2 Aspects algorithmiques

Posons tout d'abord les notations suivantes :

- Soit k un entier non négatif représentant le compteur d'itérations.
- Soit M_k le *treillis* à l'itération k.
- Soit $x_k \in \Omega \cap M_k$ le *centre de sonde* à l'itération k.
- Soit Δ_k^m le *paramètre de taille du treillis* à l'itération k.
- Soit Δ_k^p le *paramètre de taille de sonde* à l'itération k.
- Soit D un ensemble de directions générant positivement $\mathbb{R}^n$.
- Soit D_k l'*ensemble des directions de sonde* à l'itération k, construite par une combinaison entière non négative des colonnes de D.
- Soit P_k le *voisinage de sonde* à l'itération k.
- Soit S_k l'*ensemble de recherche* à l'itération k.
- Soit f_Ω la fonction objectif avec barrière, où Ω est le domaine réalisable.

L'algorithme MADS (Algorithme 1) est décrit en utilisant les précédentes notations.

Treillis Le *treillis*, sous-ensemble discret de $\mathbb{R}^n$ dont la finesse est paramétrée par Δ_k^m, est défini ainsi :

$$M_k := \left\{ x_k + \Delta_k^m D z : z \in Z_+^{|D|} \right\}$$

où Z_+ est l'ensemble des entiers non négatifs.

Étape de recherche La fonction objectif avec barrière f_Ω est évaluée en un nombre fini de points sur le treillis courant M_k, avec l'objectif d'obtenir un y faisant décroître f. Cette étape est libre de règles, pourvu qu'elle soit d'une durée finie. Si $y \in M_k$ est trouvé satisfaisant $f(y) < f(x_k)$, alors on peut poser $x_{k+1} := y$ et aller à l'étape de mise à jour. Dans le cas contraire, on passe à l'étape de sonde.

Algorithme 1 Un algorithme MADS général

Initialisation :

Point de départ : x_0 tel que $f_\Omega(x_0) < \infty$
Base positive : D
Finesse du treillis M_0 : $\Delta_0^m > 0$
Compteur : $k \leftarrow 0$
Taille du treillis limite : $\Delta_{lim}^m > 0$

Étape de recherche (optionnelle, définie par l'utilisateur) :

Générer S_k, un ensemble fini de points d'essai sur le treillis M_k
Évaluer f_Ω sur un sous-ensemble de points de S_k

Étape de sonde :

Générer D_k
Évaluer f_Ω sur un sous-ensemble non vide des points de P_k

Mises à jour des paramètres :

Si les étapes de recherche ou sonde fournissent un point y meilleur que x_k **alors**
    $x_k \leftarrow y$
    Mettre à jour $\Delta_{k+1}^m \geq \Delta_k^m$
Sinon
    Mettre à jour $\Delta_{k+1}^m < \Delta_k^m$
Fin si
Si $\Delta_{k+1}^m < \Delta_{lim}^m$ **alors**
    STOP
Sinon
    $k \leftarrow k + 1$
    Retourner à l'étape de recherche
Fin si

Étape de sonde Nous construisons avec des combinaisons entières non négatives des colonnes de D une matrice D_k, utilisée pour bâtir le *voisinage de sonde*, défini ainsi :

$$P_k \quad := \qquad \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k \quad .$$

Cet ensemble de points constitue le voisinage P_k que l'on va évaluer, caractérisé par les directions de D_k (formant une base positive) et le treillis M_k.

Mise à jour des paramètres La mise à jour du treillis se fait ainsi : $\Delta_{k+1}^m = \tau^{\omega_k} \Delta_k^m$ pour $0 < \tau^{\omega_k} < 1$, où $\tau > 1$ est un nombre rationnel constant de l'itération, $\omega_k \leq 1$ est un entier borné inférieurement par $\omega^- \leq -1$.

Si l'étape de recherche ou de sonde produit un meilleur point, i.e., un itéré réalisable $x_{k+1} \in M_k \cap \Omega$ pour lequel $f_\Omega(x_{k+1}) < f_\Omega(x_k) \in M_k \cap \Omega$, on augmente la taille du treillis $\Delta_{k+1}^m \geq \Delta_k^m$ en prenant $\omega_k \in \{0, 1, \ldots, \omega^+\}$. On passe à l'itération suivante (retour à l'étape de recherche) en considérant le nouveau (meilleur) centre de sonde.

Sinon $f_\Omega(x_k) \leq f_\Omega(x_k + \Delta_k^m d)$ pour tout $d \in D_k$, et donc x_k est un optimum local du treillis. On diminue alors la taille du treillis $\Delta_{k+1}^m < \Delta_k^m$ en prenant $\omega_k \in \{\omega^-, \omega^- + 1, \ldots, -1\}$. On passe à l'itération suivante (aux étapes de recherche et de sonde) en conservant le même centre de sonde.

Nous allons à présent nous attaquer à l'analyse de convergence de cet algorithme.

4.2.3 Analyse de convergence de MADS

Voici dans un premier temps l'analyse de convergence inhérente au cas classique, puis nous traiterons le cas des groupes de variables à la section 4.3. Les preuves concernant ces propriétés peuvent être consultées dans les articles Audet et Dennis, J. E., Jr. (2006b) et Abramson *et al*. (2009b). Il est tout d'abord nécessaire d'introduire quelques notions propres au calcul non lisse.

Définitions d'analyse non lisse

Un fonction est dite *Lipschitz* si son taux d'accroissement est borné partout. Plus formellement :

Définition 4.2.1 *Soient I un sous-ensemble non vide de $\mathbb{R}^n$ et k un réel strictement positif. La fonction $f : I \to \mathbb{R}$ est dite* k-lipschitzienne *ou* k-Lipschitz *si :*

$$\forall (x, y) \in I^2, \|f(x) - f(y)\| \leq k\|x - y\| \ .$$

Une fonction f est dite lipschitzienne *ou* Lipschitz *s'il existe $k > 0$ tel que f est k-lipschitizienne.*

Voici maintenant la définition du cône hypertangent issue de Rockafellar (1980) :

Définition 4.2.2 *Un vecteur $v \in \mathbb{R}^n$ est dit* hypertangent *à l'ensemble $\Omega \subset \mathbb{R}^n$ au point $x \in \Omega$ s'il existe un scalaire $\epsilon > 0$ tel que :*

$$y + tw \in \Omega, \forall y \in \Omega \cap B_\epsilon(x),\ w \in B_\epsilon(v),\ \textit{et}\ 0 < t < \epsilon \ .$$

L'ensemble des vecteurs hypertangents à Ω en x est appelé le cône hypertangent à Ω en x *et se note $T_\Omega^H(x)$.*

On introduit la dérivée directionnelle généralisée de Clarke (1983) :

Définition 4.2.3 *Soit une fonction f localement Lipschitz autour d'un point $x \in \mathbb{R}^n$, un vecteur $v \in \mathbb{R}^n$ et un scalaire positif $t \in \mathbb{R}^+$. La dérivée directionnelle généralisée de f en x dans la direction v, notée $f^\circ(x; v)$, est définie ainsi :*

$$f^\circ(x; v) = \lim_{\substack{y \to x \\ t \downarrow 0}} \frac{f(y + tv) - f(y)}{t} \ .$$

Voici enfin deux définitions spécifiques aux algorithmes de recherche par motifs :

Définition 4.2.4 *Une sous-séquence d'itérés de* MADS *formée de centres de sonde infructueux,* $\{x_k\}_{k\in K}$ *pour un sous-ensemble d'indices* K*, est appelée* sous-suite raffinante *si* $\{\Delta_k^p\}_{k\in K}$ *converge vers 0.*

Définition 4.2.5 *Soit* $\hat{x}$ *la limite d'une sous-séquence raffinante convergente. Si* $\lim_{k\in L} \frac{d_k}{\|d_k\|}$ *existe pour un sous-ensemble* $L \subseteq K$ *avec des directions de sonde* $d_k \in D_k$*, et si* $x_k + \Delta_k^p d_k \in \Omega$ *pour un nombre infiniment grand de* $k \in L$*, alors la limite est appelée* direction raffinante *pour* $\hat{x}$*.*

Principaux résultats pour le cas classique

Nous notons $f^\circ(x; v)$ la dérivée généralisée directionnelle de Clarke de f en x dans la direction v, et $T_\Omega^H(x)$ le cône hypertangent en x sur le domaine Ω.

Hypothèse(s) 4.2.1

- *Un point initial* $x_0 \in X$ *est fourni.*
- *La valeur initiale de la fonction objectif* $f(x_0)$ *est finie.*
- *Tous les itérés* $\{x_k\}$ *générés par* MADS *se trouvent dans un ensemble compact.*

Théorème 4.2.2 *Théorème de convergence pour* MADS *:*

Soit $\hat{x} \in \Omega$ *la limite de centres de sonde réalisables infructueux* $\{x_k\}_{k\in K}$ *sur des treillis qui tendent à être infiniment fins. Si* f *est Lipschitz localement autour de* $\hat{x}$*, alors* $f^\circ(\hat{x}; v) \geq 0$ *pour tout* $v \in T_\Omega^H(\hat{x})$*.*

Corollaire 4.2.3 *De plus, si* f *est strictement différentiable autour de* $\hat{x}$*, et si* Ω *est régulier autour de* $\hat{x}$*, alors* $f'(\hat{x}, v) \geq 0$ *pour tout* $v \in T_\Omega^H(\hat{x})$*, c'est-à-dire que* $\hat{x}$ *est un point KKT (Karush-Kuhn-Tucker) pour* $\min_{x\in\Omega} f(x)$*.*

Enfin, notons que des résultats similaires tiennent pour les limites des centres de sonde irréalisables.

4.3 Analyse de convergence : le cas des groupes

Nous allons maintenant procéder à une analyse de convergence dans le cas général des groupes de variables, qui est une généralisation du cas classique. En effet, le cas classique représente le sous-cas où toutes les variables sont contenues dans un seul groupe et sont donc déplacées simultanément à chaque itération. Nous rappelons au lecteur que les principaux résultats de convergence pour ce dernier ont déjà été récapitulés à la section 4.2.3.

4.3.1 Se ramener au MADS classique

La première idée repose sur la tentative de se ramener au déroulement d'un algorithme MADS classique, afin d'hériter directement des propriétés de convergence évoquées précédemment.

Ajouter un groupe N à chaque itération

La technique de premier abord consiste à se baser sur l'idée du PSDMADS décrit dans Audet *et al*. (2008b), où il est question d'établir un algorithme MADS parallèle efficace pour les problèmes de boîte noire à nombreuses variables. La convergence est ici assurée par un processus fonctionnant en parallèle. Ainsi, nous pourrions ajouter systématiquement (i.e., à chaque itération) aux groupements de variables définis par l'utilisateur un groupe N contenant toutes les variables. Par ce fait, nous héritons de l'analyse de convergence théorique du cas classique.

Terminer par un MADS classique

Dans un second temps, on peut penser réaliser une optimisation heuristique avec plusieurs types de groupements successifs, puis exécuter un algorithme MADS classique pour hériter de ses propriétés de convergence.

Fusionner progressivement les groupes Enfin, on peut imaginer quelque chose de plus évolutif, c'est-à-dire que la taille des groupes augmente (les groupes fusionnent) au fur et

à mesure du déroulement de l'algorithme, pour finir avec un unique groupe composé de toutes les variables, ce qui revient au cas classique.

Premières conclusions

La première approche (avec N) paraît conceptuellement brutale, car elle conduit à l'ajout d'évaluations supplémentaires afin de conserver l'analyse de convergence classique. L'approche suivante définit un cadre algorithmique général très souple : il est possible d'exécuter n'importe quel algorithme heuristique (de durée finie) pourvu que l'on termine par l'exécution d'un algorithme MADS classique. Elle peut cependant conduire à des évaluations supplémentaires si l'heuristique n'est pas définie de façon optimale (peu efficiente ou trop longue).

Enfin, on propose une instanciation particulière élégante au sein de ce cadre : la fusion progressive des groupes, reposant sur une relaxation progressive des contraintes de groupes [2]. *A priori*, il paraît logique de "guider" le sytème au départ par l'intermédiaire des groupes, puis de lui ajouter progressivement des degrés de liberté pour lui permettre de se raffiner localement. Nous avons testé ces approches (voir section 5.3) et les premiers résultats tendent à corroborer cette hypothèse et à montrer que les deux dernières approches sont meilleures que la première et le cas classique.

4.3.2 Utiliser uniquement les groupes

Nous allons ici mener l'analyse de convergence de MADS dans le cas des groupes, sans essayer de se ramener au cas classique (à l'inverse de la section précédente 4.3.1). Cependant, nous allons voir qu'il en résulte un affaiblissement des propriétés de convergence. On se place par la suite à l'étape de sonde, car c'est sur cette étape que reposent les propriétés générales de convergence des algorithmes MADS. Supposons les hypothèses 4.2.1 (cf. section 4.2.3) vérifiées et posons les notations supplémentaires suivantes :

– Soit un centre de sonde $\hat{x} \in \Omega \subset \mathbb{R}^n$, à partir duquel on construit des groupes N_q, $q \in Q$.

[2] En effet, les groupes représentent des contraintes supplémentaires (sous la forme de variables fixées) pour le système, qui doit se déplacer dans un espace réduit, comme nous allons le voir à la prochaine section 4.3.2.

– Soit une direction de sonde $v_q \in \mathbb{R}^{n_q}$ pour chaque groupe N_q.

De plus, nous pouvons compléter ce vecteur $v_q \in \mathbb{R}^{n_q}$ en un vecteur $\tilde{v}_q \in \mathbb{R}^n$ en ajoutant des zéros pour les variables $\hat{x}_i \in \overline{N_q}$.

Un groupe à la fois

Lorsque l'on modifie un seul groupe N_q à la fois selon la direction $\tilde{v}_q$, on fixe toutes les variables appartenant à $\overline{N_q} = N \backslash N_q$ sur le centre de sonde actuel $\hat{x}$. Seules les variables appartenant à N_q sont libres. En déplaçant un groupe à la fois, on travaille donc à l'itération k sur le sous-problème :

$$\min_{x \in \Omega_q(\hat{x})} f(x)$$
$$\text{où } \Omega_q(\hat{x}) = \left\{ \quad x \in \Omega \quad | \quad x_i = \hat{x}_i \quad \forall i \in \overline{N_q} \quad \right\} \ .$$

Ce sous-problème comporte $n_q = |N_q|$ variables libres. Il s'agit d'une projection du problème global sur un espace à n_q dimensions. L'ensemble $\Omega_q(\hat{x})$ est appelé *sous-domaine de Ω propre au groupe N_q au point $\hat{x}$*. Considérons le sous-espace dans lequel va travailler l'algorithme à itération fixée (c'est-à-dire l'ensemble des directions de sonde possibles pour le groupe N_q) :

$$F_q(\hat{x}) = \left\{ \quad x \in \mathbb{R}^n \quad | \quad x_i = \hat{x}_i \quad \forall i \in \overline{N_q} \quad \right\} \ .$$

Par convention, nous noterons F_q le sous-espace vectoriel $F_q(\mathcal{O})$, où $\mathcal{O}$ est l'origine de $\mathbb{R}^n$.

Assertion 4.3.1 *Tout point de sonde $\hat{x}$ généré par un groupe N_q à itération k donnée est à l'intérieur d'un sous-espace affine de dimension n_q, passant par $\hat{x}$, dont la direction est le sous-espace vectoriel F_q.*

Assertion 4.3.2 *Pour tout $\hat{x} \in \mathbb{R}^n,\ F_q(\hat{x}) \cap \Omega = \Omega_q(\hat{x})$.*

Lemme 4.3.3 *Soit $N_q \subseteq N$ un groupe de variables, et Ω_q le sous-domaine précédemment défini. Si seul le groupe N_q est autorisé à varier à chaque itération, l'algorithme* MADS *converge vers un point $\hat{x}$ satisfaisant :*

$$f^\circ_\Omega(\hat{x}; v) \geq 0, \forall v \in T^H_{\Omega_q(\hat{x})}(\hat{x})$$

où $T^H_{\Omega_q(\hat{x})}(\hat{x})$ est le cône hypertangent à $\Omega_q(\hat{x})$ en $\hat{x}$.

Preuve. Par le théorème 4.2.2, l'algorithme MADS est assuré de converger vers un tel point (stationnaire au sens de Clarke dans le sous-espace). ■

On retrouve en corrolaire, lorsque $N_q = N$, le théorème de l'algorithme MADS classique (car alors $\Omega_q(\hat{x}) = \Omega$). Cependant, si $N_q \neq N$, $F_q(\hat{x})$ ne forme au maximum qu'un hyperplan de $\mathbb{R}^n$ (ceci est le cas lorsque N_q contient $n-1$ variables), ce qui est loin de garantir qu'il n'existe aucune direction $v' \in T^H_\Omega(\hat{x})$ diminuant la fonction objectif, c'est-à-dire telle que $f^\circ_\Omega(\hat{x}; v') < 0$.

Généralisation

Sans perte de généralité et par souci de simplification, supposons par la suite que Q est l'ensemble (non vide) des indices des groupes que l'on peut modifier à toutes les itérations et que le centre de sonde $\hat{x}$ coïncide avec l'origine. Ces hypothèses ne sont pas restrictives : la première permet d'éviter l'introduction d'indices propres aux itérations ; grâce à la seconde, nul besoin de distinguer *sous-espace affine* passant par $\hat{x}$ de direction F_q et *sous-espace vectoriel* F_q, puis tous deux se confondent. Enfin, étant donnés les résultats de la section précédente, posons les notations suivantes :

- Soit F_Q l'union des sous espaces vectoriels F_q, $q \in Q$.
- Soit H_i l'hyperplan de $\mathbb{R}^n$ orthogonal au i-ème vecteur de la base canonique.

Assertion 4.3.4 *Le nombre de façons de créer un groupe avec n variables est fini et vaut 2^n.*

Assertion 4.3.5 *Soit $q_1, q_2 \in Q$. Si $N_{q_1} \subseteq N_{q_2}$, alors $F_{q_1} \subseteq F_{q_2}$.*

Assertion 4.3.6 *Tout point de sonde $\hat{x}$, généré par les groupes N_q, $q \in Q$, est à l'intérieur de F_Q, l'union de sous-espaces F_q, $q \in Q$.*

Proposition 4.3.7 *Le sous-espace vectoriel F_{q_i} associé au groupe contenant toutes les variables sauf celle d'indice i est égal à l'hyperplan H_i.*

Preuve. Soit $H_i \in \mathbb{R}^n$ l'unique hyperplan de $\mathbb{R}^n$ orthogonal au i-ème vecteur de la base canonique. Par contradiction, on peut montrer que H_i contient exactement toutes les directions ne modifiant pas la i-ème coordonnée. Par définition, il en est de même pour l'espace F_{q_i} où $N_{q_i} = N - \{i\}$. Donc $H_i = F_{q_i}$. ■

Proposition 4.3.8 *Supposons que $N_q \neq N$ pour tout $q \in Q$. Alors l'union des sous-espaces vectoriels $\bigcup_{q \in Q} F_q$ est incluse dans l'union de n hyperplans $\bigcup_{i \in N} H_i$.*

Preuve. Les plus grands groupes ne formant pas de recouvrement contiennent $n - 1$ variables, et ont donc une unique variable qui reste fixe. Cette famille de groupes, qui contient exactement n éléments différents, est notée $\{N_{q_i}\}_{i \in N}$, où chaque groupe d'indice q_i ne contient pas la i-ème variable. À chaque groupe N_{q_i} est associé son sous-espace vectoriel F_{q_i}. Donc, puisque tous les groupes possibles ne contenant pas la i-ème variable sont inclus dans N_{q_i}, par l'assertion 4.3.5 et la proposition 4.3.7, on arrive à :

$$\bigcup_{q \in Q} F_q \subseteq \bigcup_{i \in I} F_{q_i} = \bigcup_{i \in I} H_i \ . \ \blacksquare$$

Ainsi, tous les groupements possibles sans recouvrement ne permettent de visiter au maximum que n hyperplans orthogonaux car chacun des espaces vectoriels engendrés par un groupe $N_q \neq N$ est contenu dans l'un des éléments de la famille $\{H_i\}_{i \in N}$. On est donc bien loin d'être en mesure d'explorer l'ensemble $\mathbb{R}^n$.

Lorsque l'on crée des points de sonde pour chacun des groupes N_q, $q \in Q$, à l'itération k, on génère $|P_k|$ points de sonde dans des directions $w \in F_Q$.

Théorème 4.3.9 *Soit un ensemble de groupes N_q, $q \in Q$. Soit $\hat{x} \in \Omega$ la limite de centres de sonde réalisables infructueux $\{x_k\}_{k \in K}$ sur des treillis qui tendent à être infiniment fins. Si f est Lipschitz localement autour de $\hat{x}$, en modifiant tous les groupes de façon individuelle à chaque itération, l'algorithme* MADS *converge vers un point $\hat{x}$ vérifiant :*

$$f^\circ_\Omega(\hat{x}; w) \geq 0, \forall w \in \bigcup_{q \in Q} T^H_{\Omega_q(\hat{x})}(\hat{x}) \ .$$

Preuve. En appliquant le lemme 4.3.3 à tous les groupes N_q, $q \in Q$, on montre que l'algorithme MADS converge vers un point satisfaisant :

$$f_\Omega^\circ(\hat{x}; w) \geq 0, \forall w \in T_{\Omega_q(\hat{x})}^H(\hat{x}), \forall q \in Q \ .$$

En considérant l'ensemble $\bigcup_{q \in Q} T_{\Omega_q(\hat{x})}^H(\hat{x})$, on aboutit à l'expression du théorème. ■

Nous retrouvons, en corollaire, le résultat du théorème 4.2.2 : si l'un des groupes N_q forment un recouvrement de N (MADS classique), alors $T_{\Omega_q}^H(\hat{x}) = T_\Omega^H(\hat{x})$. Mentionnons que l'ajout à l'algorithme MADS classique (formant un groupe N) de groupes plus petits peut renforcer les propriétés de convergence, puisqu'il est possible (voir Figure 4.2) d'avoir :

$$T_\Omega^H(\hat{x}) \subset \bigcup_{q \in Q} T_{\Omega_q(\hat{x})}^H(\hat{x}) \ .$$

Cela vient du fait que la non-convexité du domaine dans le voisinage du point limite diminue drastiquement le cône hypertangent (parfois jusqu'à l'ensemble vide). Lorsqu'on crée des groupes, on ne considère plus qu'une fraction du domaine. Éventuellement, l'intersection entre le sous-espace et le voisinage réalisable peut alors être convexe, rendant non vide le cône hypertangent au sous-domaine.

Notons également que, lorsque $N_q \neq N$, $\forall q \in Q$, l'algorithme converger vers une sorte d'équilibre local, possiblement sous-optimal, où aucun groupe n'a intérêt à subir de variation (infinitésimale) si tant est que les autres groupes conservent leur position actuelle. Cependant, encore ici, cela ne garantit pas qu'il n'existe pas une direction $w' \in T_\Omega^H(\hat{x})$ telle que $f_\Omega^\circ(\hat{x}; w') < 0$.

4.3.3 Conclusions sur la convergence

En fait, en créant des groupes de variables, l'algorithme est contraint d'utiliser moins de possibilités de directions qu'il n'en a dans le cas classique. D'après la section 4.3.2, nous pouvons déduire de façon générale qu'un groupe ne formant pas un recouvrement de N ne permet de visiter au maximum qu'un hyperplan de $\mathbb{R}^n$. De plus, la proposition 4.3.8

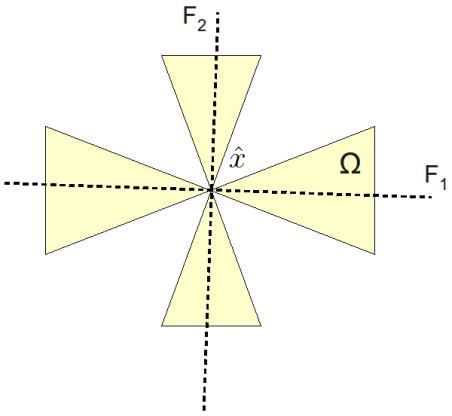

FIGURE 4.2 Soit $\Omega \subset \mathbb{R}^2$ l'union des quatre aires triangulaires (fermées) apparentes. Nous avons $N = \{1, 2\}$. Créons deux groupes additionnels $N_1 = \{1\}$ et $N_2 = \{2\}$, et supposons que l'algorithme MADS converge vers le point $\hat{x}$ situé à l'intersection des cônes. Le cône hypertangent $T^H_\Omega(\hat{x})$ est vide, tandis que l'union des cônes hypertangents $T^H_{\Omega_q(\hat{x})}(\hat{x})$, $\forall q \in \{1, 2\}$ est formée de deux segments ouverts, chacun correspondant au domaine $\Omega_q = \Omega \cap F_q$ privé de ses extrémités.

ramène à n hyperplans orthogonaux l'union de tous les sous-espaces explorables par tous les groupes possibles dans le cas sans recouvrement total. On pourra voir par analogie avec les cas $2D$ et $3D$ que cela est relativement peu : deux droites dans un plan, ou trois plans dans l'espace. Par ce fait, il est certain que nous ne parviendrons pas à obtenir une analyse de convergence équivalente au cas classique : en effet, nous ne pouvons aspirer à construire un espace de directions normalisées dense dans la sphère unité avec seulement l'union d'un nombre fini d'hyperplans.

Pour conclure, tant qu'il n'existe pas, à partir d'un certain nombre fini d'itérations, un groupe formant un recouvrement de N appelé à chaque itération, nous ne retombons pas dans le cas de l'analyse de convergence classique de MADS définie à la section 4.2.3. C'est pourquoi nous allons chercher à nous y ramener par des méthodes de groupements dynamiques, qui seront abordées à la section 4.4.2.

4.4 Discussion générale sur la gestion des groupes

À la lumière des précédentes conclusions, nous allons maintenant voir les différentes approches pour la gestion des groupes. Il s'agit de bâtir une stratégie pour se ramener au cas classique (où toutes les variables sont dans un seul groupe) de façon intelligente.

4.4.1 Gestion statique

NOMAD *3.1* gère les groupes de façon *statique*, i.e., la création des groupes se fait uniquement au début de son lancement, et il conserve ce groupement tout au long de l'algorithme.

Exemple 4.4.1 *Soit un vecteur de variables $x = (x_i)_{1\leq i\leq 6}$. Si nous définissons trois groupes $N_1 = (1, 2, 3)$, $N_2 = (3, 4)$ et $N_3 = (5, 6)$,* NOMAD *va générer à chaque itération des directions pour chaque groupe. Il va générer des directions pour N_1, puis il passera à N_2, et enfin, ce sera au tour de N_3. Ensuite,* NOMAD *évaluera la liste des points de sonde (constituée ici de 14 points si on utilise une base positive maximale), dans un certain ordre (à définir) et se déplacera selon les résultats. Finalement, il bouclera en recommençant,*

et ceci jusqu'à vérification d'un critère d'arrêt. Evidemment, puisque dans le cas général où aucun groupe ne forme un recouvrement de N, l'union des directions normalisées est loin d'être dense dans la sphère unité, il en est de même pour cet exemple de groupement statique. En effet, nous n'explorons jamais l'espace $\mathbb{R}^6$ en modifiant (entre autres) x_1 et x_6 ensemble.

Cette approche, qui a le mérite de sa simplicité, manque de flexibilité. En effet, imaginons que l'on définisse un ensemble de groupes propres à un sous-domaine commun, chaque groupe correspondant à un objet. On peut alors rencontrer le problème suivant : les objets forment une configuration sous-optimale, mais, si l'on déplace les groupes individuellement, on dégrade la solution actuelle : tous les groupes sont "piégés" dans un minimum local de leur sous-espace. Ainsi, avec un groupement statique, la configuration ne peut évoluer, tel qu'illustré à la Figure 4.3.

De fait, s'il est empiriquement intéressant de constituer des groupes de variables de faible taille (par exemple en les groupant par objets) pour accélérer la progression de l'algorithme, il n'en demeure pas moins nécessaire de se débloquer des situations où la création de groupes entrave cette dernière. La solution réside dans le compromis : ne pas bloquer trop de degrés de liberté dès le début pour ne pas empêcher l'algorithme de progresser, mais en bloquer suffisamment pour diriger l'algorithme vers une amélioration significative en peu d'itérations. Ainsi, il est nécessaire, pour se débloquer des situations sous-optimales, de constituer, modifier et détruire dynamiquement les groupes.

4.4.2 Gestion dynamique

Nous souhaitons raffiner l'approche précédente en proposant de modifier les groupes dynamiquement tel qu'on le laissait entrevoir dans la section 4.3.1, afin d'explorer le plus d'espace possible tout en conservant une certaine cohésion dans notre démarche d'utilisation des groupes.

Exemple 4.4.2 *Par exemple, dans l'algorithme* PSDMADS *défini dans Audet et al. (2008b), un processus maître confie un groupe de variables (donc un sous-domaine à explorer) à chacun de ses processus esclaves. Dès lors que l'un d'eux a terminé sa tâche, le maître*

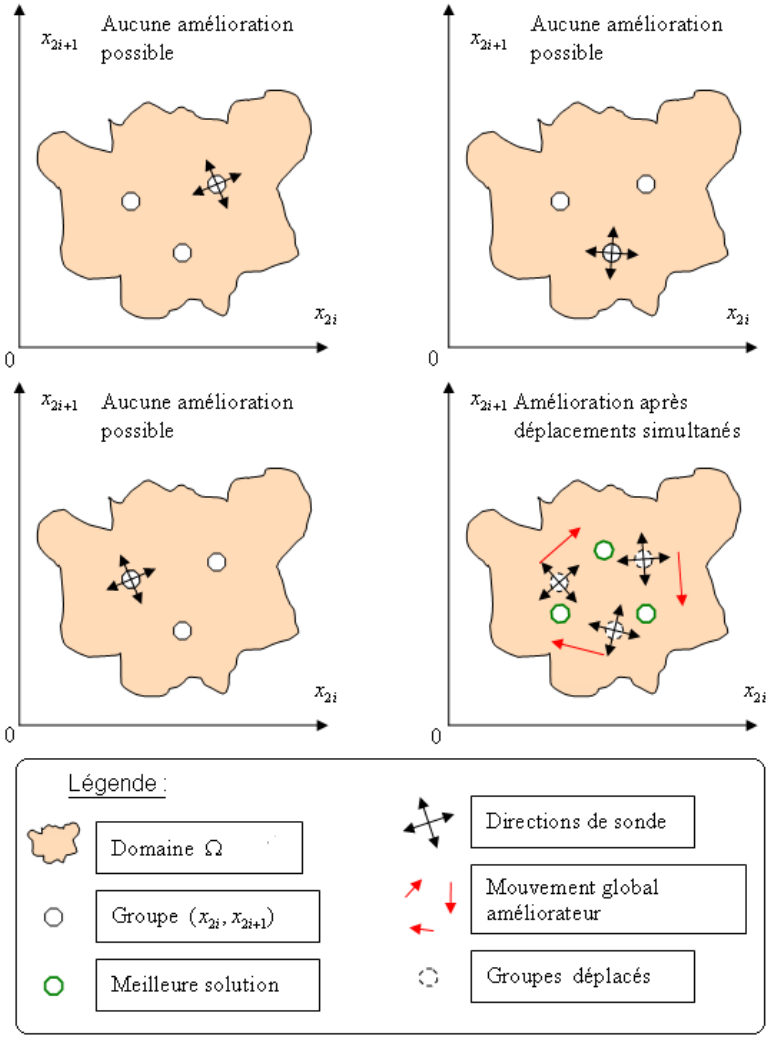

FIGURE 4.3 Mouvement global requis : aucune amélioration de la valeur objectif $f(x)$ n'est possible si l'on ne déplace qu'un seul groupe à la fois.

redéfinit le groupe associé à l'esclave pour le relancer sur un nouveau domaine. Les esclaves fonctionnent donc de manière asynchrone sur des sous-domaines, et seul un processus esclave particulier tournant en parallèle assure l'exploration (de façon théorique) dans toutes les directions de l'espace $\mathbb{R}^n$.

Au niveau conceptuel, la gestion dynamique des groupes va se faire selon l'Algorithme 2, qui se décompose en quatre étapes principales : l'initialisation, l'exécution, la vérification et la reconfiguration.

Algorithme 2 Gestion dynamique des groupes idéale

1: Initialisation :
   – Création des groupes, critère d'arrêt, X_0
2: Exécution :
   – Lancement de NOMAD avec les paramètres définis à l'étape précédente.
3: Vérification :
   – Analyse de la structure des groupes, du critère d'arrêt de l'étape précédente
   – Si on a atteint notre but, alors STOP, sinon aller à l'étape 4
4: Reconfiguration :
   – Opérations sur les groupes
   – Changement du critère d'arrêt, de X_0, de la taille du treillis
   – Retour à étape 2

On rappelle que pour hériter de l'analyse de convergence théorique du cas classique, il ne peut y avoir qu'un nombre fini d'itérations où aucun groupe ne forme un recouvrement de l'ensemble des variables. Cela revient à imposer la condition pratique suivante : *"À partir d'un certain nombre fini d'itérations, un groupe contient toutes les variables."*

Ainsi, nous avons deux possibilités pour construire les variantes d'algorithmes possibles :

– Une gestion terminant par un MADS classique (Algorithme 3).
– Une gestion avec groupe N à chaque itération (Algorithme 4).

Cependant, pour l'Algorithme 4, déterminer quand créer le groupe N et combien de points de sonde sont à générer est un problème en soi : il s'agit d'un compromis entre restreindre l'espace exploré (k grand, p petit) et augmenter le nombre d'évaluations (k petit, p grand). Nous nous limiterons, pour cette version, au cas où $p = 2n$ et $k = 0$.

Algorithme 3 Gestion générale en terminant par un MADS classique

1: Gérer les variables selon une heuristique quelconque jusqu'à ce que l'un des groupes contienne toutes les variables
2: Lancer une exécution "classique" à partir de là

Algorithme 4 Gestion générale avec groupe N

1: Gérer les variables selon une heuristique quelconque jusqu'à un certain critère d'arrêt
2: Après $k \geq 0$ itérations, créer un groupe N contenant toutes les variables, qui génèrera $p > 0$ point(s) de sonde supplémentaires à chaque itération

Enfin, nous allons comparer les performances des algorithmes issus des deux structures. Cependant, pour plus de facilité d'implémentation concernant la vérification, on peut supposer que l'algorithme est capable d'identifier s'il s'agit ou non de la dernière exécution *avant* de l'effectuer. On peut alors utiliser un booléen indiquant si oui (1) ou non (0), il s'agit de la dernière exécution de NOMAD. Ainsi, on concentre les difficultés sur la reconfiguration. Nous allons nous employer à décrire ce nouvel algorithme, baptisé GM-MADS (*Group Management with MADS*), dont une instanciation pratique sera décrite à la section 5.

4.4.3 Les outils d'analyse

Afin de gérer les groupes de variables de façon logique, il est nécessaire de les analyser. Nous pourrions effectuer des calculs spécifiques sur les objets concrets, tels que la distance inter-objets, leurs déplacements relatifs, ou encore leur voisinage, afin d'envisager, par exemple, des regroupements par proximité ou mobilité. Ou bien, on peut privilégier une approche plus abstraite en réalisant des statistiques sur les données recueillies au fur et à mesure du déroulement de l'algorithme, de façon à dégager des liens ou une relation d'ordre entre les variables.

Les avantages des calculs spécifiques aux objets concrets sont qu'ils sont aisés à mettre en oeuvre et ne nécessitent que peu de calculs supplémentaires. L'analyse de sensibilité, quant à elle, est une approche plus générale, mais qui nécessite souvent d'établir des hypothèses sur les interactions entre les variables pour être efficace, e.g. indépendance,

normalité, linéarité ou monotonie. De plus, dès lors que l'on souhaite obtenir des résultats sophistiqués, les démarches deviennent très vite lourdes à mettre en place (calcul matriciel, détermination des modèles et coefficients de régression, décomposition de variance).

4.4.4 Comment créer des groupements concrètement ?

Avec les calculs spécifiques

L'objectif est d'identifier, lorsqu'on traite avec des variables décrivant des objets, une manière d'exploiter directement les données sans passer par un procédé général d'analyse de sensibilité. Par exemple, lorsque plusieurs objets sont définis sur un sous-domaine commun, on peut former des groupes de variables pour lesquels les distances, voisinages et mouvements relatifs ont un sens concret : peut-on utiliser ces grandeurs pour bâtir un algorithme de groupement fructueux ? Peut-on exploiter la notion de voisinage de chacun des objets qui constituent x_k ou vaut-il mieux conserver la notion de voisinage globale ? Remarquons également que si le terrain sur lequel sont définis les objets est fragmenté, il est difficile de déterminer un point $y \in \Omega$ réalisable dans le voisinage de x_k en modifiant la position de tous les objets simultanément. D'où la solution de déplacer un sous-ensemble des objets à la fois et donc la nécessité de définir un voisinage pour chacun des objets.

Par distance Si l'idée d'un groupement par objets semble naturelle, on peut y ajouter un groupement par distance reposant sur l'hypothèse implicite que deux objets "proches" interagissent entre eux. Ce n'est pas toujours le cas, y compris dans un contexte de positionnement : par exemple, en télécommunications, deux antennes peu distantes mais de fréquences suffisamment différentes ne s'influencent pas mutuellement. Cependant, dans de nombreuses applications, la distance constitue une mesure cohérente des interactions, et c'est notamment le cas dans notre problème de localisation : les déplacement groupés peuvent alors permettre de se débloquer d'une situation sous-optimale.

Par absence de mouvement Le groupement par immobilité repose quant à lui sur l'hypothèse qu'un objet deumeurant fixe est vraisemblablement proche de sa position optimale (pour son sous-espace). Il peut alors être intéressant de se concentrer davantage sur les

autres objets mobiles, en groupant les objets mouvants séparément et les objets immobiles ensemble. Non seulement, cela donne une priorité de déplacement aux objets mouvants (donc vraissemblablement loin de leur position d'équilibre), mais cela permet également aux objets immobiles d'être déplacés ensemble (et donc d'accroitre l'ensemble des directions possibles), afin de les débloquer de situations sous-optimales.

Avec l'analyse de sensibilité

On souhaite déterminer quels sont les variables ou les groupes de variables importants et quand est-ce qu'ils le sont. Ceci peut permettre à la fois d'ordonner les candidats à l'évaluation de façon pertinente et de créer des groupements judicieux.

Grouper les variables a posteriori L'idée serait ici de mener dans un premier temps une analyse de sensibilité sur les dernières itérations, afin d'identifier les variables actuellement prépondérantes (i.e., celles qui modifient significativement la fonction objectif et les contraintes relaxables au regard des autres variables) et de constituer des groupements pertinents (Algorithme 5).

Algorithme 5 Former des groupes *a posteriori*

1: Déterminer une relation d'ordre entre les variables
2: Former des groupes à partir de celles-ci

Créer des petits groupes avec les variables importantes Les règles de ce groupement sont simples :

- Créer des groupes de petites tailles avec les variables importantes.
- Créer des groupes de plus grande taille avec les variables restantes.

En effet, il semble judicieux de déplacer en priorité les variables actuellement influentes, au risque de délaisser momentanément les autres. En créant des groupes de petite taille avec une variable importante, on est sûr de déplacer cette variable significativement au cours des prochaines itérations (Figure 4.4), ce qui peut permettre d'accélerer la progression de l'algorithme.

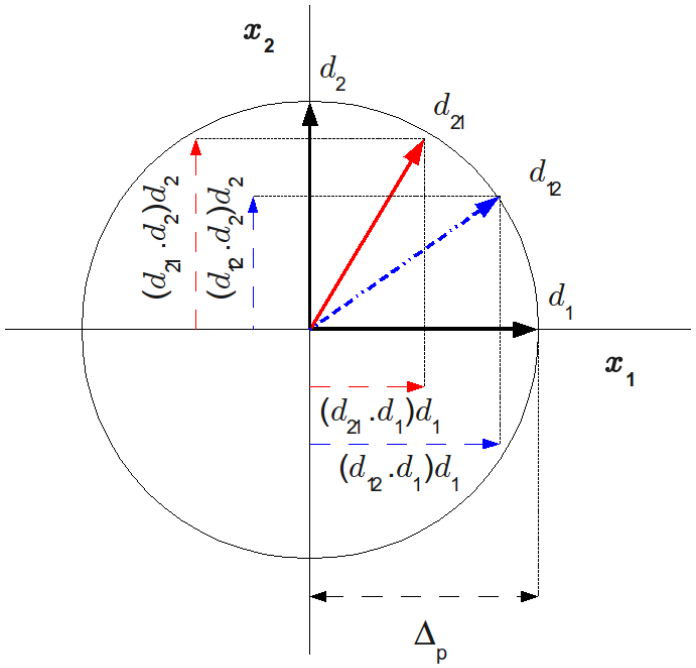

FIGURE 4.4 Directions de sonde d_1, d_2, d_{12}, d_{21} pour une itération donnée sur un exemple à deux variables (x_1, x_2), où Δ_p est le paramètre de sonde. En créant un groupe de petite taille, on fait varier plus significativement (en moyenne) chaque variable du groupe le long de son axe, que si on crée un groupe de grande taille.

Modèles pour les fonctions de score

On souhaite estimer une ou des fonctions caractérisant l'influence des variables contenues dans un groupe et/ou de ce groupe sur la variation de la fonction objectif de façon à pouvoir, d'une part, ordonner éventuellement les candidats à évaluer et d'autre part, grouper les variables de manière adéquate.

Soit un groupe de variables N_q (N_q peut être éventuellement composé d'une unique variable). Soit un échantillon d'évaluations S. De chaque élément de S, on peut extraire entre autres les valeurs des variables x_s, la valeur de la fonction objectif $f(x_s)$, ainsi que la valeur de la violation des contraintes relaxables $h(x_s)$ (à valeur dans $\mathbb{R}+$).

Une fonction d'influence U assigne à un couple (N_q, S) une valeur réelle ou *utilité*. On pourra alors établir, pour une fonction d'influence donnée, une relation d'ordre entre l'ensemble des groupes possibles, puis les classer, et décider de les gérer en conséquence. Par exemple, un groupe N_q peut être considéré comme fructueux pour un échantillon S, s'il fournit, sans dégrader ou en dégradant légèrement $h(x_k)$, beaucoup de points diminuant la valeur courante $f(x_k)$, ou encore un ou plusieurs points diminuant significativement la valeur courante $f(x_k)$.

Gestion des échecs Lorsque nous souhaitons effectuer des statistiques sur les valeurs de fonctions (objectifs, contraintes et variables), il se pose le choix de prendre en compte ou non les échecs d'évaluation dus au domaine X défini par les contraintes non relaxables. Cependant, en les considérant, la différence d'amélioration entre un point irréalisable et un point réalisable sera quasi-infinie, ce qui aura pour effet de mettre en avant les groupes qui ont des voisinages largement réalisables, et non les groupes qui ont des voisinages fructueux. L'ajout d'une pénalité pour les points $x \notin X$ pose le problème de son estimation numérique. Nous avons donc évincé cette difficulté en ne considérant que les points appartenant à X.

Une question de mémoire Il est à noter que si un groupe de variables est fructueux ou influent à un moment donné dans l'algorithme, il se peut très bien qu'il ne le soit plus quelques itérations plus tard. Il faut certes tenir compte des évènements passés pour établir des statistiques pertinentes, mais faut-il effectuer les statistiques sur l'ensemble

des évaluations passées ou seulement sur la dernière exécution ? Nous ne considérerons que les points obtenus au cours de l'exécution précédente pour éviter que les premières évaluations (dont les améliorations vont forcément être significatives puisque le système n'est en général pas proche d'une configuration optimale) n'influencent définitivement l'ensemble des futures configurations.

Régression linéaire multiple Pour chaque point x_s de l'échantillon S, on suppose que la valeur de la fonction objectif est une combinaison linéaire des variables d'entrée :

$$f(x_s) = \sum_{i \in N} a_i x_{i,s} + b + \epsilon_s$$

où ϵ_s est l'erreur empirique associée au point x_s. On recherche les coefficients a_i, b tels qu'il minimisent globalement la somme des carrés des valeurs absolues des erreurs :

$$\min_{a_i,b} \sum_{x_s \in S} \epsilon_s^2 = \min_{a_i,b} \sum_{x_s \in S} |f(x_s) - b - \sum_{i \in N} a_i x_{i,s}|^2 \quad .$$

Ainsi, on peut voir l'influence relative des variables sous une hypothèse (probablement fausse) de relation linéaire. Bien que le modèle soit trop simple pour modéliser fidèlement la réalité, cela nous donne cependant des indications précieuses sur les variables à influence forte et faible. Enfin, si le coefficient de corrélation sous l'hypothèse linéaire est trop faible, nous pouvons opérer une transformation des rangs en vue d'améliorer la robustesse du modèle étant donné que les relations sont non linéaires. Introduite originellement par Iman et Conover (1979), une telle méthode permet, selon Jacques (2005), d'affaiblir l'hypothèse de linéarité en la remplaçant par la monotonie de la fonction objectif par rapport à chaque variable.

4.4.5 Réalisabilité : le cas du domaine fragmenté

Nous avons vu à la section 3.2.1 un exemple d'applications où le domaine de définition Ω est hautement fragmenté. Dans de tels cas, il est intéressant (lorsque cela est possible) de modifier la méthode d'optimisation pour assurer son efficacité, notamment par le biais d'un

prétraitement. En effet, l'irréalisabilité, d'un point de vue algorithmique, ne pose pas de problème particulier à MADS : après avoir évalué le candidat y non réalisable, il posera $f(y) = +\infty$ (dans un contexte de minimisation) et changera de candidat. Cependant, si Ω est très fragmenté, il est très probable qu'un grand nombre de points d'essai testés soient déclarés irréalisables et que MADS amorce autour de la solution de départ une sous-suite raffinante. Au final, cela risque d'entraver la progression de l'algorithme ou même de le faire échouer à se déplacer de sa position initiale.

Tout d'abord, avant d'évaluer un candidat, s'il est possible de tester sa réalisabilité à faible coût, il est judicieux de le faire à l'avance, afin d'économiser une évaluation de boîte noire peu édifiante. Bien sûr, il existe des cas où aucune information n'est connue *a priori* sur le domaine X défini par les contraintes non relaxables, auquel cas on procèdera directement à l'évaluation sans prétraitement possible. Cependant, nous allons nous intéresser ici au cas où il est possible de tester de façon non coûteuse (avant l'évaluation de f et des fonctions de contraintes relaxables c_j) l'appartenance de y à X.

Notons qu'il faudra dans tous les cas adapter son prétraitement au domaine : lorsqu'un point d'essai est déclaré irréalisable, si cela est possible et relativement non coûteux, on peut tenter de rechercher dans son voisinage un point réalisable. Pour conserver la logique MADS, l'idéal serait d'être capable de déterminer, pour tout $y \in M_k$, le point $\bar{y} \in M_k$ réalisable le plus proche.

Il est *a priori* difficile de trouver un point réalisable dans un domaine fragmenté en n dimensions car cela impose de tester un ensemble de points qui croit exponentiellement avec le nombre de variables. On ne peut aspirer à décrire exhaustivement comment déterminer un point réalisable voisin dans un espace à n dimensions, car cela dépend de la présence de sous-domaines et de la densité de l'espace Ω dans $\mathbb{R}^n$. Il est donc nécessaire d'adapter une telle recherche au problème. Dans le contexte de variables de positionnement, nous travaillons souvent avec des groupes propres à un sous-domaine commun de faible dimension : il semble alors naturel de chercher à assurer la réalisabilité d'un point en assurant tout d'abord celle de chacune des positions des groupes.

Il y a deux façons de voir les choses en pratique : soit on suppose intégrée la recherche d'un point réalisable au sein de la boîte noire, soit on essaie d'intégrer la recherche au sein du cadre algorithmique de MADS.

Intégration de la recherche dans la boîte noire Le plus simple est certainement de supposer directement intégrée la recherche d'un point réalisable au sein de la boîte noire, éventuellement en confiant au client la tâche de bâtir un prétraitement déterministe en fonction du problème considéré. Ainsi, d'un point de vue optimiseur, à un point évalué va correspondre une unique valeur de fonction objectif, et on retombe alors dans le cas usuel.

Intégration de la recherche dans l'étape de recherche Nous pouvons également considérer que le prétraitement est contrôlé par l'optimiseur : assurons-nous simplement de sa cohérence théorique dans le cadre de l'algorithme MADS.

On va supposer que chaque évaluation est précédée d'un traitement déterministe de la forme de l'Algorithme 6. Ce dernier ne modifie que les points irréalisables en tentant de leur associer un point réalisable quelconque. Lorsqu'un tel point est trouvé, le point d'essai original est remplaçé par ce dernier. Dans le cas contraire, l'algorithme déclare simplement un échec et efface le point d'essai initial de liste des évaluations à effectuer. Il peut être au premier abord frappant de constater que l'on peut tout aussi bien effacer un point irréalisable ou encore le remplacer par un point réalisable quelconque, sans dégrader l'analyse de convergence théorique. Cela est dû à la souplesse du cadre algorithmique de MADS, puisqu'on peut considérer chaque évaluation de ce type (appartenant éventuellement à l'étape de sonde) comme appartenant à l'étape de recherche. En effet, tous les points modifiés étant initialement des échecs de toute façon, l'étape de sonde est bel et bien effectuée entièrement.

Toutefois, si l'on hérite en théorie de l'analyse de convergence classique, on peut remarquer que, lorsque le domaine réalisable est très fragmenté, le cône hypertangent sera souvent vide, rendant peu significatifs les théorèmes précédemment évoqués.

De plus, pour limiter le nombre de tests de réalisabilité, on peut imaginer borner notre recherche à un périmètre fini de largeur Δ_k^s. Dans cette dernière approche, on ne déconnecte pas la recherche d'un point réalisable de la logique de MADS. En effet, si l'on substitut à chaque fois un point (trop) éloigné au point y qui devait être initialement testé, la notion de voisinage "proche" propre aux algorithmes de descente locale n'est pas respectée. Toutefois, en pratique, un prétraitement sur une large portion de domaine peut avoir un rôle diversificateur intéressant en déterminant des points d'essai prometteurs éloignés, comme

Algorithme 6 Prétraitement pour corriger l'irréalisabilité (Cas pratique)

$L \leftarrow P_k \cup S_k$
Pour chaque $y \in L$ **faire**
  Si $y \notin \Omega$ **alors**
    Essayer de trouver $\bar{y} \in M_k \cap \Omega$
    Si Succès **alors**
      $y \leftarrow \bar{y}$
    Sinon
      $L \leftarrow L \backslash \{y\}$
    Fin si
  Fin si
Fin pour

en optimisation globale.

On peut imaginer enfin compliquer un peu le tout, en appliquant la même logique au prétraitement qu'à l'étape de sonde, c'est-à-dire en imposant au point testé au cours d'une étape de sonde d'être à l'intérieur d'un périmètre qui tendra vers 0 si notre centre de sonde est un élément d'un suite raffinante. Cependant, le problème serait que les valeurs objectif des points évoluent avec le déroulement de l'algorithme. C'est-à-dire qu'à un point donné peuvent correspondre deux valeurs objectifs : finie lorsque le périmètre de recherche du prétraitement permet de trouver un point voisin réalisable, et infinie sinon. L'algorithme n'est alors plus déterministe et complique notablement l'analyse de convergence. Enfin, il semble évident qu'une recherche dans un voisinage trop petit n'apporterait techniquement rien, puisqu'aucun point réalisable voisin ne sera déterminé.

Seul le cas particulier où il existe un sous-domaine commun à deux dimensions sera traité ici de manière pratique (à la section 5).

Chapitre 5

UNE IMPLÉMENTATION PRATIQUE DE GM-MADS

Nous allons traiter ici un problème particulier de localisation de balises sur un sous-domaine commun fragmenté en deux dimensions, sans contraintes relaxables. Cette boîte noire artificielle a été conçue en tant que substitut du problème original de l'IREQ. En effet, l'accès à la boîte noire originale étant limité, nous avons été contraint de recourir à ce modèle (simplifié) afin de réaliser une batterie de tests préliminaires. L'algorithme que nous proposons va illustrer plusieurs principes : *prétraiter la réalisabilité* afin d'éviter que la fragmentation du domaine entrave la progression, *bâtir des groupements évolutifs* afin de générer des directions d'exploration fructueuses et *ordonner les candidats* pour l'évaluation dans une optique opportuniste. Les choix pour construire les groupements interviennent dans notre cas à trois niveaux : un groupement initial, un algorithme de reconfiguration et enfin une règle de groupement secondaire.

5.1 Aspects combinatoires

Comme le laissait entendre le chapitre précédent, le problème de déterminer une technique de groupement optimale est très fortement combinatoire, et ceci pour plusieurs raisons. Tout d'abord, le problème de déterminer, à groupement fixé, une solution optimale à l'une des instances considérées est déjà en lui-même combinatoire. Ensuite, le nombre de groupes de variables (non vides) possibles pour un ensemble de n éléments est égal à $2^n - 1$, et ces groupes de variables peuvent se fusionner, se diviser, se recombiner à peu près n'importe comment. Se limiter aux cas où l'on traite uniquement avec des groupes formant une partition de l'ensemble des variables n'arrange pas énormément les choses,

puisque le nombre de partitions possibles d'un ensemble de cardinalité n, égal au nombre de Bell, est potentiellement gigantesque. En voici l'expression par la formule de Dobinski :

$$B_n = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!} \ .$$

Par exemple, pour à peine quinze variables, on peut créer $1\ 382\ 958\ 545$ partitions ! Enfin, ajoutons à cela la diversité des réglages des paramètres algorithmiques, ainsi que le nombre d'instances, et nous obtenons, dans une perspective exhaustive, une quantité de tests défiant toute raison.

Il est donc nécessaire de restreindre ces travaux à quelques cas particuliers que l'on juge pertinents. Rappelons que le principal objet de ce travail est de prouver ou réfuter l'hypothèse selon laquelle l'utilisation des groupes de variables peut être bénéfique. Dans le cas positif, nous proposerons une ou des manières de grouper les variables efficacement pour les instances considérées. Déterminer une façon optimale, même à instance fixée, est un défi non trivial qui sort du cadre de ce mémoire.

5.2 La boîte noire : substitut pour les balises GMON

Cette boîte noire a été bâtie dans le but de modéliser, en première approximation, le comportement des balises GMON (Gamma-MONitoring) qui servent à l'interpolation par géostatistique du manteau neigeux, problème qui a déjà été présenté au chapitre 3. Les variables impliquées dans ce problème sont des variables de positionnement.

5.2.1 Le domaine

Le domaine où les variables peuvent être placées est représenté par un ensemble de points sur une carte en deux dimensions. Les différentes composantes physiques font en sorte que le domaine est largement fragmenté. De par cette fragmentation, l'application directe d'une méthode d'optimisation (que ce soit MADS ou une autre) est vouée à l'échec. Étant donné que la carte définissant le problème nous est connue, nous avons pu modifier l'algorithme pour que les points d'essai soient déplacés afin de coïncider avec un

point réalisable avoisinant. Cette projection sur le domaine, décrite plus en détail à la section 5.3.5, n'est pas coûteuse étant donné que le bassin est représentable dans un espace à deux dimensions.

5.2.2 La fonction objectif

L'information dont nous disposons est incomplète. En particulier, nous n'avons pas accès au logiciel interne à l'IREQ permettant de calculer les mesures d'erreurs associées à un positionnement de GMON. Pour contourner cette difficulté, nous avons conçu une fonction substitut, en accord avec le cadre défini par Booker *et al*. (1999).

Nous avons utilisé, pour la modélisation de l'erreur de mesure, une fonction potentielle substitut décroissante avec la distance, dont voici l'expression :

$$f(x) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \frac{1}{\sqrt{e_{ij}(x)}}$$

avec (N_i, N_j) les dimensions de la carte et $x \in \mathbb{N}^{2g}$ un vecteur comportant les positions de g GMON. La valeur $e_{ij}(x)$ correspond à la fiabilité de la mesure au point (i, j) en fonction d'une configuration de GMON. Cette valeur varie entre 0 (pire erreur) et 1 (aucune erreur) et est calculée selon :

$$e_{ij}(x) = \begin{cases} 1 & \text{si } d_{ij}(x) \geq 1, \\ d_{ij}(x) & \text{si } 0.030 \leq d_{ij}(x) < 1, \\ (d_{ij}(x))^{1.2} & \text{si } 0.025 \leq d_{ij}(x) < 0.03, \\ (d_{ij}(x))^{1.5} & \text{si } 0.020 \leq d_{ij}(x) < 0.025, \\ (d_{ij}(x))^{2} & \text{si } d_{ij}(x) < 0.020, \end{cases}$$

avec

$$d_{ij}(x) = \sum_{\ell=0}^{g-1} c_{ij}(x_{2\ell}, x_{2\ell+1})$$

et

$$c_{ij}(x_u, x_v) = \begin{cases} 1 & \text{si } x_u = i \text{ et } x_v = j, \\ \frac{0.8}{\sqrt{(x_u - i)^2 + (x_v - j)^2}} & \text{sinon.} \end{cases}$$

Ansi, la fiabilité brute spécifique à la balise (x_u, x_v) au pixel (i, j), notée $c_{ij}(x_u, x_v)$, varie directement inversement proportionellement à la distance à la balise. On somme ensuite sur l'ensemble des GMON pour obtenir la fiabilité brute totale $d_{ij}(x)$. Ensuite, on recadre les valeurs trop grandes à 1, puis on met les $d_{ij}(x)$ à la puissance $\beta \in \{1, 1.2, 1.5, 2\}$, ce qui nous permet d'obtenir les valeurs de fiabilité finales $e_{ij}(x)$ (voir Figure 5.1). L'erreur locale pour une configuration x au pixel (i, j) est égale à l'inverse de la racine carrée de la fiabilité $e_{ij}(x)$: un pixel possède donc une valeur d'erreur locale minimale lorsqu'il a une fiabilité maximale, c'est-à-dire lorsqu'il coïncide avec la position d'une balise GMON (ce qui est cohérent). Enfin, on somme sur l'ensemble des pixels de la carte l'erreur pour obtenir l'erreur totale $f(x)$. Les valeurs des constantes ont été ajustées afin d'obtenir des solutions visuellement satisfaisantes, qui ont été ensuite validées par des experts de l'IREQ.

5.3 Ensemble de problèmes tests

Nous cherchons ici à définir un banc d'essai de référence : différentes méthodes de groupements de variables seront produites, puis testées sur un ensemble d'instances donné. Leurs performances seront ensuite comparées et analysées. Cela nous permettra de sélectionner les meilleurs algorithmes en vue de résoudre efficacement le problème original issu de l'IREQ.

5.3.1 Les instances

Les domaines sont définis à partir de cartes de bassins québecois fournies par l'IREQ. Nous en considérerons trois, respectivement nommés *Gatineau* (GAT), *Saint-Maurice* (STM) et *La Grande* (LG). Ces domaines (Figure 5.2) sont de différentes densités : GAT est très fragmenté, tandis que LG l'est peu et STM est intermédiaire.

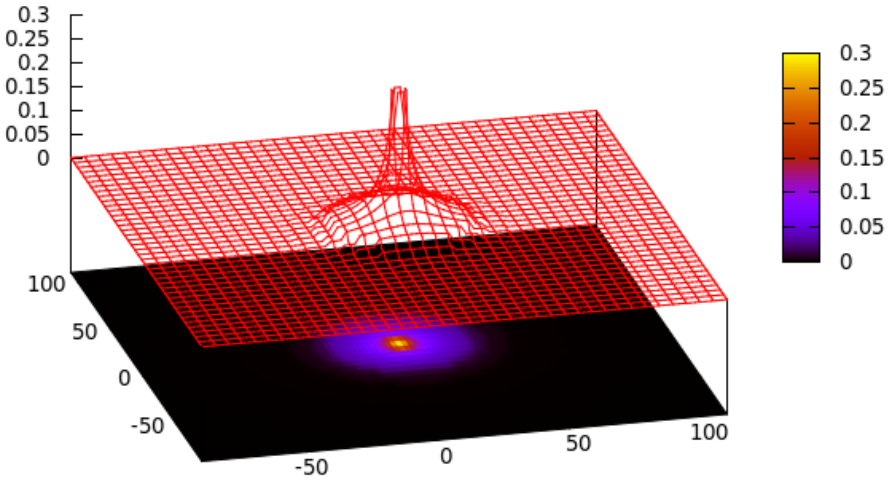

FIGURE 5.1 Valeurs de fiabilité en fonction de la position : le centre du puits de potentiel caractérise la position de l'unique balise GMON.

Le nombre de GMON que nous utilisons varie de 5 à 10 (10 à 20 variables). Tous les GMON, sauf mention contraire, sont supposés libres.

Cela fait un total de 18 instances à tester pour chaque méthode de groupement.

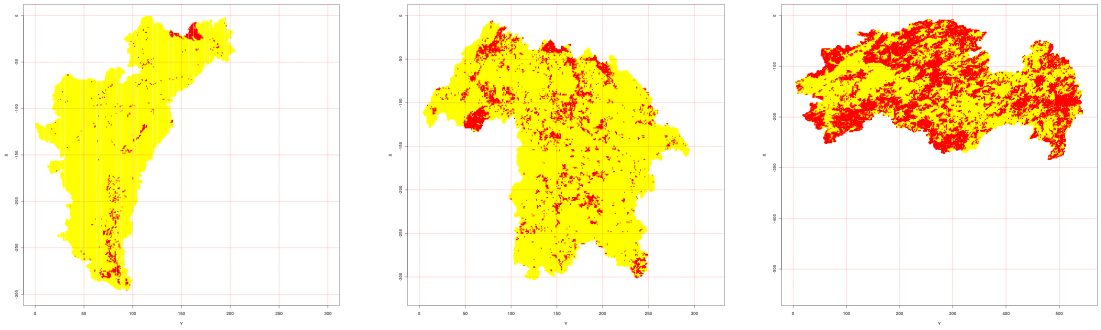

FIGURE 5.2 Les trois cartes utilisées (de gauche à droite respectivement GAT, STM et LG) : on a représenté la zone globale où est calculée l'erreur (en clair), sur laquelle on a superposé le sous-domaine commun réalisable pour les balises GMON (en foncé).

5.3.2 L'initialisation

Nous allons ici présenter tous les critères initiaux choisis, à savoir la solution initiale, le groupement initial et les instances considérées.

Solution initiale

Nous avons testé différentes approches dont la génération aléatoire de solutions initiales (pratique car automatisable). Toutefois, cette dernière empêche les résultats d'être réellement significatifs dans la mesure où améliorer une solution aléatoire est "facile". De plus, cela risque de biaiser les comparaisons et de permettre aux reconfigurations "violentes" d'être davantage performantes à court terme, alors même qu'elle ne seront peut-être pas efficaces sur les instances réelles qui disposent déjà de solutions initiales de bonne qualité. Pour pallier ces défauts, nous choisirons pour solutions initiales des configurations faites "à la main" de façon *relativement* pertinente, à raison d'une pour chaque instance.

Groupement initial

Les *groupements initiaux* peuvent être choisis parmi les partitions suivantes :

- I : Variable individuelle (n groupes d'une seule variable).
- X : Paires par GMON (x_{2i}, x_{2i+1}) en coordonnées cartésiennes.
- P : Paires de coordonnées alternées (décrite ci-dessous).
- A : Abscisses et ordonnées (2 groupes de $n/2$ variables).
- T : Toutes ensemble (un seul groupe de n variables).

Lorsque le nombre d'objets est pair, le groupement P groupe les variables (x_{2i}, x_{2i+2}), (x_{2i+1}, x_{2i+3}). Dans le cas impair, P groupe la première et la dernière variable, et reproduit, pour le nombre de variables restantes (formant un nombre pair de paires), le groupement propre au cas pair.

I déplace les balises (individuellement) uniquement verticalement ou horizontalement. X modifie la position d'une seule balise à la fois (il s'agit de l'approche la plus naturelle). P déplace simultanément deux balises différentes selon chacune un axe vertical ou horizontal. A affecte les positions de la moitié des balises, selon leur axe vertical, puis selon leur axe horizontal. T modifie l'ensemble des variables, donc toutes les positions des balises : il s'agit du MADS classique.

Les groupements I et X ont été conçus en exploitant la structure du problème. Nous nous attendons à ce qu'ils produisent de meilleurs résultats que A et P, qui ont été conçus arbitrairement, à titre comparatif.

Critère d'arrêt

Afin de faire décroître le plus rapidement possible la valeur de la fonction objectif, on aimerait pouvoir reconfigurer les groupes lorsque l'algorithme peine à trouver de nouvelles solutions. De plus, il est nécessaire, pour des raisons pratiques, de limiter le temps de calcul total.

Les critères suivants sont disponibles :

1. *Nombre d'échecs successifs* : Lorsqu'un point est plusieurs fois d'affilée le centre de sonde et donc vraissemblablement un élément d'une suite raffinante, on reconfigure.

2. *Nombre d'évaluations* : Au bout d'un certain nombre d'évaluations.
3. *Nombre d'itérations* : Au bout d'un certain nombre d'itérations.
4. *Temps de calcul* : Au bout d'un certain temps.
5. *Valeur limite de la taille du treillis* : Lorsque la taille de treillis passe en dessous d'un certain seuil.

Le critère 1 renseigne sur l'incapacité de l'algorithme à améliorer (même légèrement) le candidat actuel. Fixer une valeur évoluant avec le nombre d'exécutions (de façon croissante) permet d'assurer des reconfigurations régulières tout en prenant en compte la progression algorithmique.

Les critères 2 et 4 sont habituellement dictés par les limitations pratiques du budget. Comme souvent, les critères 2, 3 et 4 comportent le désavantage du choix explicite de paramètres influents (nombres d'évaluations, itérations ou temps).

Le critère 5 est celui qui peut permettre d'assurer la convergence théorique puisque, par construction, la taille du treillis va tendre vers zéro lorsqu'on se trouve en un point vérifiant des conditions d'optimalité de premier ordre. Il peut être également utilisé pour identifier des échecs de façon subtile, avec des règles du type : *"Si la taille du treillis passe en dessous de sa valeur initiale, on reconfigure."*

Nous limiterons nos travaux en choisissant un critère d'arrêt total basé sur le nombre d'évaluations, indépendant de l'instance, fixé à 1000 évaluations de boîtes noires maximum sur l'ensemble de l'algorithme. Cette valeur est raisonnable en termes de temps de calcul, et permet de révéler de façon pertinente l'évolution des différents algorithmes, à la fois sur le problème de l'IREQ et sur notre modèle substitut.

Paramètres algorithmiques de NOMAD

Nous choisissons de définir la graine de Halton comme le n-ième nombre premier, ce qui permet, selon Abramson *et al*. (2009b), de décorréler les directions de sonde. La taille du treillis initiale Δ_0^m est multidimensionnelle et dépend de la taille de la carte considérée : ses composantes associées aux variables d'indice pair et d'indice impair sont respectivement fixées au dixième de la longueur et la largeur maximales du domaine concerné. Dans toute la suite, nous supposons, sauf mention contraire, que la recherche est *opportuniste*,

c'est-à-dire que dès lors qu'un candidat y meilleur que le centre de sonde actuel x_k est découvert, l'itération courante k s'achève, et une nouvelle itération commence avec pour centre de sonde le point y. Ces trois choix correspondent au valeurs par défaut du logiciel NOMAD.

5.3.3 La reconfiguration

La reconfiguration est à l'origine basée sur la fusion progressive des groupes de variables. On peut raffiner cette vision, en donnant aux groupes la possibilité de se diviser ou d'être conservés. Rappelons que pour conserver intacte la convergence théorique, les critères doivent être tels, qu'une séquence d'itérés raffinante (i.e., engendrée par une suite d'échecs à partir d'un centre de sonde), doit conduire à terme à la fusion totale des variables (dans un seul groupe).

On souhaite établir quand et comment réaliser la reconfiguration. Pour ce faire, on cherche à exhiber :

- Des critères d'arrêt, dépendants ou non de la progression de l'algorithme (voir section 5.3.2).
- Des fonctions d'influence, caractérisant l'efficacité actuelle ou le potentiel futur de tel point ou tel groupe de variables sur la progression de l'algorithme.
- Des critères, éventuellement basés sur les fonctions d'influence, servant à décider quels groupes conserver, fusionner ou diviser.

Ainsi, à chaque début d'étape de reconfiguration, on pourra déterminer une relation d'ordre entre les points et/ou les groupes et décider lesquels prédominent. On va se concentrer par la suite sur les fonctions d'influence et les critères de fusion, division et conservation qui y sont associés.

Quand reconfigurer ?

Les reconfigurations doivent être assez fréquentes pour avoir une utilité, mais ne doivent pas non plus entraver la progression de l'algorithme. Tous les critères d'arrêt présentés à la section 5.3.2 son envisageables, pourvu qu'ils soient correctement ajustés. Cependant, afin, d'une part, de limiter le nombre de tests à effectuer, et d'autre part, de générer des résultats

exploitables pour le problème concret de l'IREQ, nous nous consacrerons dans la suite des tests à un unique critère d'arrêt, portant sur le *nombre d'échecs consécutifs* :

– EC : Reconfigurer après r itérations non fructueuses consécutives, r étant le numéro de l'exécution actuelle (initialisé à 1 à la première exécution).

Ce critère a été choisi après des tests portant sur le nombre d'évaluations, sur la valeur de la taille de treillis et le nombre d'échecs successifs. Il est relativement souple dans la mesure où il fait intervenir les performances de l'algorithme sans avoir à fixer à l'avance des paramètres arbitraires prépondérants.

Comment reconfigurer ?

Un ensemble de règles de fusion, division et conservation qui forment un tout cohérent est appelé *algorithme de reconfiguration*. Voici un récapitulatif des *algorithmes de reconfiguration* qui ont été utilisés, décrits plus précisément à la section 5.3.4 :

– S : Statique.
– D : Dynamique avec fusion par distance.
– K : Dynamique par distance avec une méthode de classe *K-means*.
– M : Dynamique avec gestion par mouvement.
– F : Dynamique avec fusion par mouvement.
– V : Dynamique par sensibilité avec une méthode de type régression linéaire multiple.
– W : Dynamique par sensibilité avec une méthode de type régression linéaire multiple avec transformation des rangs.

Règle secondaire Certains algorithmes de reconfiguration ne groupent pas directement l'ensemble des variables. C'est-à-dire qu'ils agissent uniquement sur un sous-ensemble des variables. Les *règles secondaires* servent à définir comment gérer les variables restantes. Nous distinguons trois règles possibles (la lettre "R" est ajoutée pour différencier les *règles secondaires* des *groupements initiaux*) :

- XR : Paires par GMON (x_{2i}, x_{2i+1}) en coordonnées cartésiennes.
- IR : Variable individuelle (un groupe par variable).
- TR : Toutes ensemble.

Continuité entre les exécutions Chaque étape d'exécution (indicée par r) influence l'étape de reconfiguration qui suit. La meilleure solution courante sert de point de départ à l'exécution suivante. L'index de Halton est systématiquement incrémenté d'une unité à partir de sa valeur à la fin de l'exécution précédente, de façon à décorréler les directions d'exploration testées sur l'ensemble des exécutions. Le treillis est mis à jour de façon logique, sa taille étant recalculée à partir de la valeur courante de l'indice du treillis. En fonction de l'*algorithme de reconfiguration* utilisé, l'état des groupes et le critère d'arrêt sont éventuellement modifiés.

5.3.4 Les algorithmes de reconfiguration

Toutes les méthodes de groupement sont initialisées de manière semblable (Algorithme 7). Un fois spécifiés l'instance, le groupement initial, la règle secondaire de groupement par défaut et le critère d'arrêt pour la reconfiguration, il ne reste plus qu'un seul choix (le plus important) : l'*algorithme de reconfiguration*. Nous allons nous employer ici à décrire ceux que nous avons utilisés. Rappelons que les idées sous-jacentes à ces algorithmes ont déjà été explicitées à la section 4.4.4.

Utiliser la distance

Les algorithmes par distance se base sur l'hypothèse que, dans notre contexte, les objets peu distants vont davantage interagir entre eux, et que la modification simultanée d'un ensemble d'objets proches pourrait permettre de débloquer des situations sous-optimales.

Fusionner les objets proches avec mémoire (D) Le groupe G, initialement vide, va progressivement grossir en absorbant un objet supplémentaire à chaque exécution (Algorithme 8). Cet objet est choisi en tant que plus proche de l'ensemble G selon une distance euclidienne. Lorsque tous les objets ont été absorbés, on effectue une dernière exécution,

Algorithme 7 Initialisation commune

1: Carte : $c \in \{GAT, STM, LG\}$
2: Nombre de GMON : $g \in \{5, 6, \ldots, 10\}$
3: Groupes initiaux créés : $t \in \{I, X, P, A, T\}$
4: Règle secondaire : $l \in \{IR, XR, TR\}$
5: Critère d'arrêt pour reconfiguration : $CR \leftarrow EC$ (échecs consécutifs)
6: Nombre maximum d'évaluations total : $CT \leftarrow 1000$
7: Point de départ : $x_0 \leftarrow x_0(c, g)$
8: Taille du treillis initiale : $\Delta_0^m \leftarrow \Delta_0^m(c)$
9: Approche $opportuniste$ activée
10: Nombre de variables : $n \leftarrow 2g$
11: Graine de Halton : $H \leftarrow H(n)$
12: Algorithme MADS (types de directions) : $a \leftarrow$ ORTHOMADS $2n$

avec toutes les variables groupées ensemble (MADS classique). L'algorithme effectuera au maximum g séquences.

Partitionner l'ensemble des objets par proximité (K) Identifier les objets qui sont proches peut se faire par classification automatique (*clustering*) en utilisant un algorithme de type *K-means*. Il s'agit de découper en K classes (*clusters*) un ensemble d'éléments, de façon à minimiser la somme sur l'ensemble des éléments des carrés des distances de chaque élément au centre de gravité de sa partition. Ce problème, en termes de complexité, est prouvé NP-complet même avec seulement deux classes par Aloise *et al*. (2009).

D'un point de vue pratique, nous utilisons la fonction *kmeans()* (Hartigan et Wong (1979)) du logiciel libre de statistiques **R** (R Development Core Team (2010)) en faisant des appels à l'interpréteur **R** au milieu du code *C++* de NOMAD, grâce à la fonction *system()*. Le nombre de classes, dans notre cas, varie au cours du déroulement de l'algorithme. Ce dernier est initialisé avec g classes, puis diminue à chaque nouvelle exécution le nombre de classes, ce qui permet de converger vers une fusion de l'ensemble des variables. Ce nombre doit bien entendu être entier, mais n'importe quelle fonction décroissante en escalier associant au numéro de l'exécution un nombre de classes convient. Pour rester simple, nous diminuerons ce nombre incrémentalement à chaque nouvelle exécution, ce qui fait converger en g exécutions maximum en une fusion complète (Figure 5.3).

Algorithme 8 Fusion par distance avec mémoire (D)

Initialisation :

1: $G \leftarrow \emptyset$

Étape 1 :

1: Exécuter a (ORTHOMADS) jusqu'à vérification de CR ou CT
2: **Si** $G = \emptyset$ **alors**
3: Ajouter à G les variables associées aux deux objets les plus proches
4: **Sinon**
5: Ajouter à G les variables associées à l'item le plus proche de G
6: **Fin si**
7: **Si** G forme un recouvrement de l'ensemble des variables **alors**
8: Aller à l'étape 2
9: **Sinon**
10: Former un groupe avec l'ensemble des variables de G
11: Grouper les autres variables selon la règle secondaire
12: **Fin si**
13: Aller à l'étape 1

Étape 2 :

1: Former un groupe avec l'ensemble des variables de N
2: Exécuter a jusqu'à vérification de CT

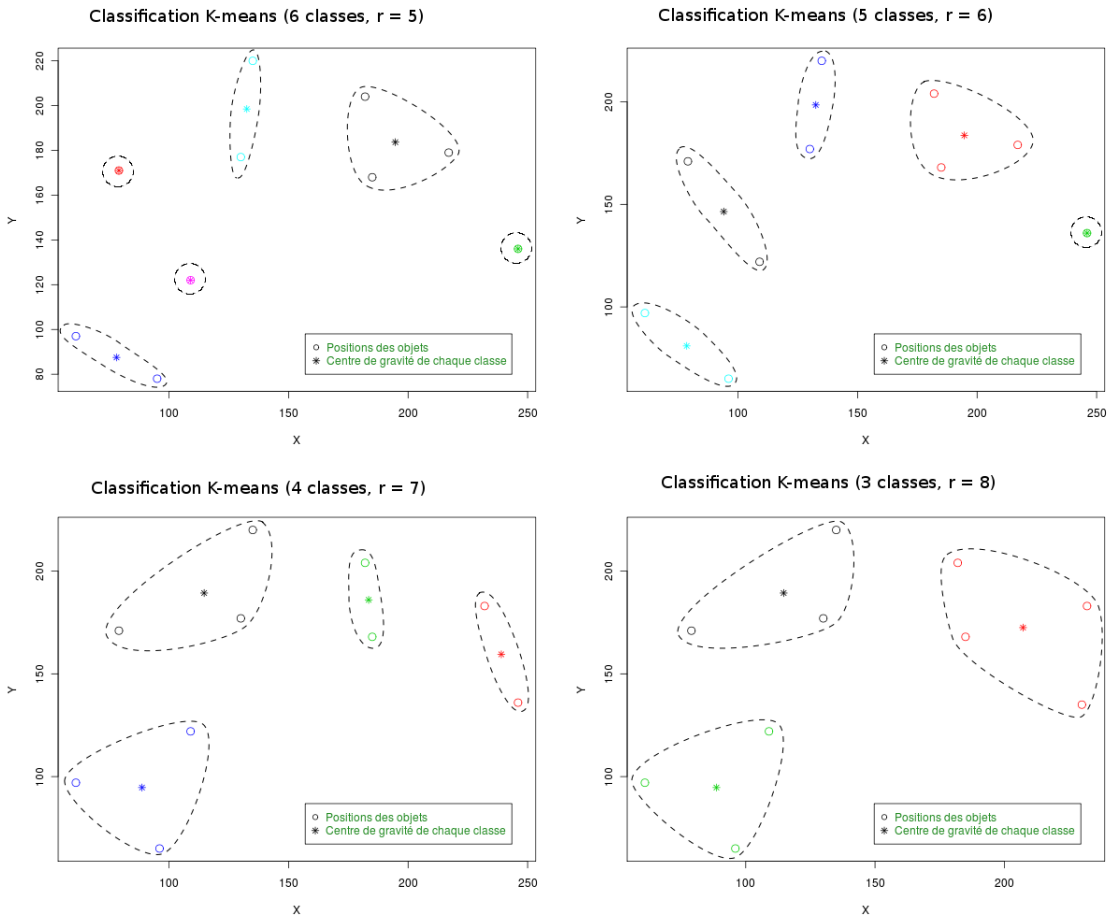

FIGURE 5.3 Fusion par l'algorithme *K-means* avec 10 objets : les objets proches sont regroupés en classes et le nombre de classes est incrémentalement diminué à chaque nouvelle exécution r.

Utiliser l'immobilité

On stocke la solution de départ x_r^{start} au début de l'exécution r, puis on la compare avec la solution obtenue x_r^{end} à la fin de l'exécution r. La reconfiguration agit alors comme suit pour chaque objet g_i : si la position de g_i a évolué, ses variables sont candidates à former un groupe ; sinon, le groupe est candidat à la fusion (explicitée ci-après). Tel qu'évoqué à la section 4.4.4, un objet ne se déplaçant pas est très problablement proche de sa position optimale, et il peut alors être intéressant de se concentrer prioritairement sur les autres candidats. En groupant les objets immobiles ensemble et les autres par groupes de plus petites tailles, on modifie principalement ces derniers. Deux variantes de cette solution ont été étudiées : avec mémoire et sans mémoire.

Tout fusionner sans mémoire (M) Elle découle simplement des règles précédemment définies, les objets candidats à la fusion sont fusionnés ensemble, les objets candidats à former un groupe forment un groupe (Algorithme 9). Cela converge vers une fusion complète car, à terme, la configuration des objets va s'approcher d'un optimum local, et donc il arrivera un instant où aucun objet ne parviendra à se déplacer. Le nombre d'exécutions pour parvenir à une fusion complète ne peut être contrôlé à l'avance.

Tout fusionner avec mémoire (F) Les objets candidats à la fusion ou qui ont été candidats à la fusion sont fusionnés ensemble, et seuls les objets candidats à former un groupe qui n'ont jamais été fusionnés forment un groupe. Ainsi, une entité croît en absorbant à chaque exécution les objets immobiles. Il s'agit d'une simple variante de (M) où l'on ne réinitialise pas G entre chaque exécution. Cela peut converger très rapidement vers une fusion complète des variables.

Utiliser l'analyse de sensibilité

On se propose ici de définir des méthodes tentant de déterminer et d'exploiter l'influence des variables sur la valeur de la fonction objectif, en analysant l'ensemble de l'échantillon généré au cours de la précédente exécution.

Algorithme 9 Fusion par mouvement sans mémoire (M)

Initialisation :

1: $G \leftarrow \emptyset$

Étape 1 :

1: Exécuter a (ORTHOMADS) jusqu'à vérification de CR ou CT
2: $G \leftarrow \emptyset$
3: Ajouter à G les variables associées aux objets qui ne se sont pas déplacés
4: **Si** G forme un recouvrement de l'ensemble des variables **alors**
5: Aller à l'étape 2
6: **Sinon**
7: Former un groupe avec l'ensemble des variables de G
8: Grouper les autres variables selon la règle secondaire
9: **Fin si**
10: Aller à l'étape 1

Étape 2 :

1: Former un groupe avec l'ensemble des variables de N
2: Exécuter a jusqu'à vérification de CT

Régression linéaire multiple *a posteriori* (V) Nous proposons un modèle théorique de régression linéaire multiple, qui déjà été décrit à la section 4.4.4, en considérant uniquement pour échantillon les évaluations effectuées au cours de l'exécution précédente. Nous utilisons le paradigme *a posteriori* (Algorithme 5). Nous intègrons à notre code *C++* un appel à la fonction *lm()* du logiciel **R** (R Development Core Team (2010)) de manière semblable à la section 5.3.4. Les variables sont classées par ordre décroissant de valeur absolue de coefficient de régression. Les premières variables sont considérées prioritaires et groupées individuellement, tandis que les autres sont groupées toutes ensemble. À chaque nouvelle exécution, l'ensemble des variables prioritaires est diminué de deux éléments, ce qui a pour effet d'aboutir à la fusion complète des variables en g exécutions (Algorithme 10).

Régression linéaire multiple *a posteriori* avec transformation des rangs (W) Nous reproduisons le traitement de l'algorithme V, en opérant une transformation des rangs, introduite par Iman et Conover (1979), sur l'ensemble des données : la fonction objectif et les valeurs des variables d'entrée sont substituées à leur rang dans l'échantillon, ce qui

Algorithme 10 Régression linéaire multiple *a posteriori* (V)

Initialisation :

1: $G \leftarrow \emptyset$
2: $z \leftarrow 0$

Étape 1 :

1: Exécuter a (ORTHOMADS) jusqu'à vérification de CR ou CT
2: $G \leftarrow \emptyset$
3: $z \leftarrow z + 2$
4: Effectuer une régression linéaire multiple sur l'échantillon d'évaluations généré au cours de la précédente exécution
5: Classer les variables par ordre décroissant de valeur absolue de coefficient de régression
6: Ajouter à G les z dernières variables
7: **Si** G forme un recouvrement de l'ensemble des variables **alors**
8: Aller à l'étape 2
9: **Sinon**
10: Former un groupe avec l'ensemble des variables de G
11: Grouper les autres variables selon la règle I
12: **Fin si**
13: Aller à l'étape 1

Étape 2 :

1: Former un groupe avec l'ensemble des variables de N
2: Exécuter a jusqu'à vérification de CT

permet théoriquement de réduire la sensibilité de la méthode à l'hypothèse linéaire.

5.3.5 Prétraitement

Réalisabilité

À la lumière de la précédente discussion (voir section 4.4.5), notre recherche de réalisabilité, bornée à un périmètre carré, s'effectue via une marche déterministe en spirale sur l'ensemble des pixels. Celle-ci dépend de la direction de sonde originale et va des points les plus proches aux plus éloignés, tel qu'illustré aux Figures 5.4 et 5.5. Sur ces dernières, on constate que la recherche bornée à 10 pixels (A) se solde dans certains cas par un échec (i.e., aucun point réalisable n'est déterminé dans le voisinage exploré), tandis que la recherche à 25 pixels (B) se solde systématiquement par un succès, éventuellement éloigné du point d'essai original. Les points déjà réalisables ne sont bien sûr pas modifiés. Après des tests portant sur les valeurs $\{5, 10, 15, 20, 25\}$, la demi-largeur limite de la spirale est finalement fixée à 15 pixels. Ce choix permet de progresser en dépit de la fragmentation du domaine tout en évitant les évaluations de points abusivement éloignés des points d'essai originaux.

Ordre d'évaluation

Nous ne classons pas les groupes, mais uniquement les points d'évaluation, selon une fonction d'utilité basée sur les valeurs de la carte d'erreur obtenue avec le centre de sonde actuel x_k. En reprenant l'expression de la fonction objectif (cf. section 5.2.2), on peut interpréter l'inverse de $\sqrt{e_{ij}(x_k)}$ comme l'erreur courante associée au pixel de coordonnées (i, j). Chaque point d'essai à évaluer y se voit attribuer un score égal à la somme des erreurs associées à chaque position de balise :

$$u(y) = \sum_{\ell=0}^{n-1} \frac{1}{\sqrt{e_{y_{2\ell}, y_{2\ell+1}}(x_k)}} \ .$$

Nous classons ainsi les points d'évaluation (par ordre d'utilité décroissante), pour privilégier les déplacements de balises vers les lieux où l'erreur courante est maximale, ce qui devrait permettre, intuitivement, de la réduire davantage.

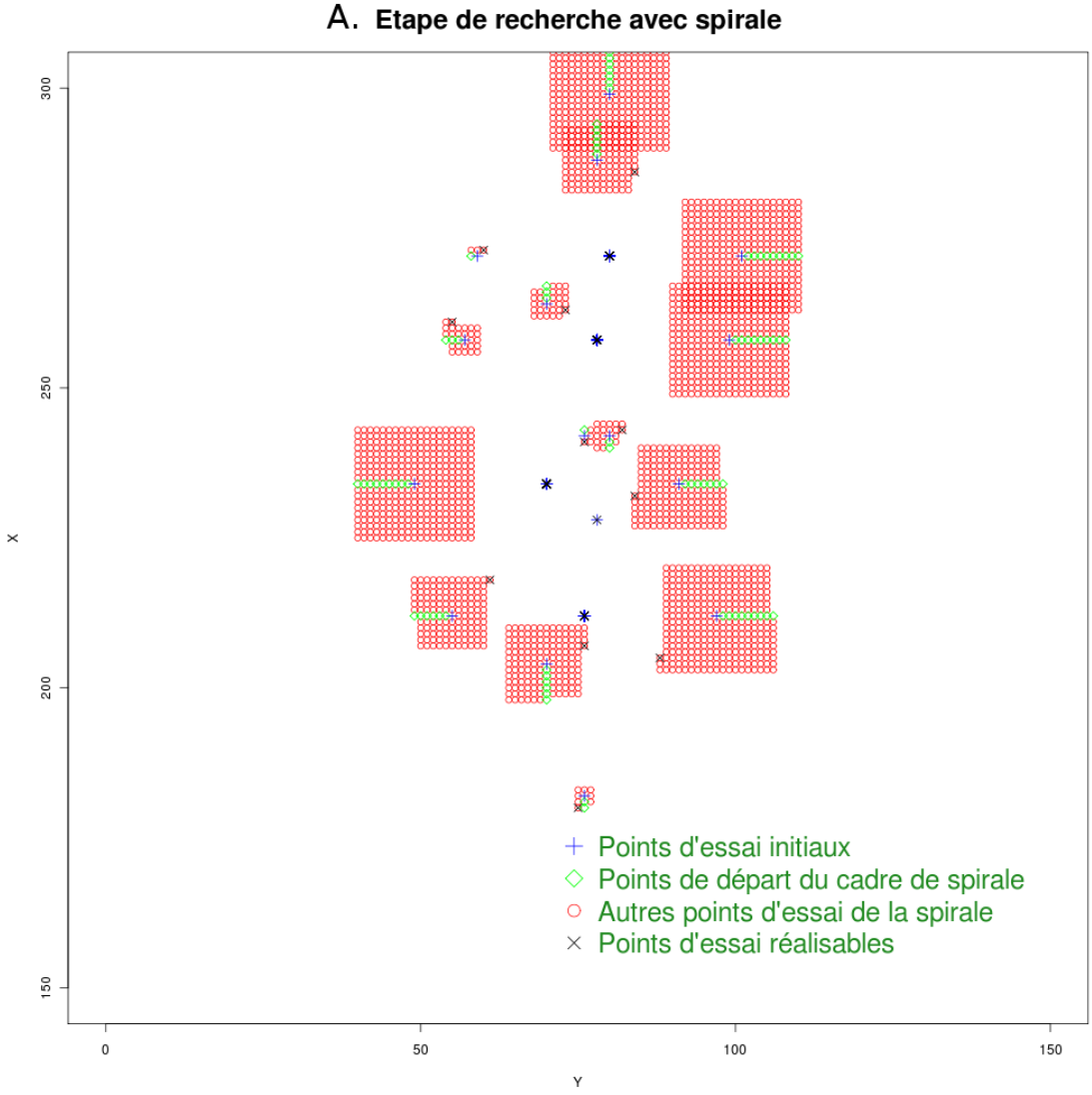

FIGURE 5.4 Une méthode de recherche en spirale pour 5 balises (donc 10 variables et 20 points d'essai), avec une demi-largeur de 10 pixels.

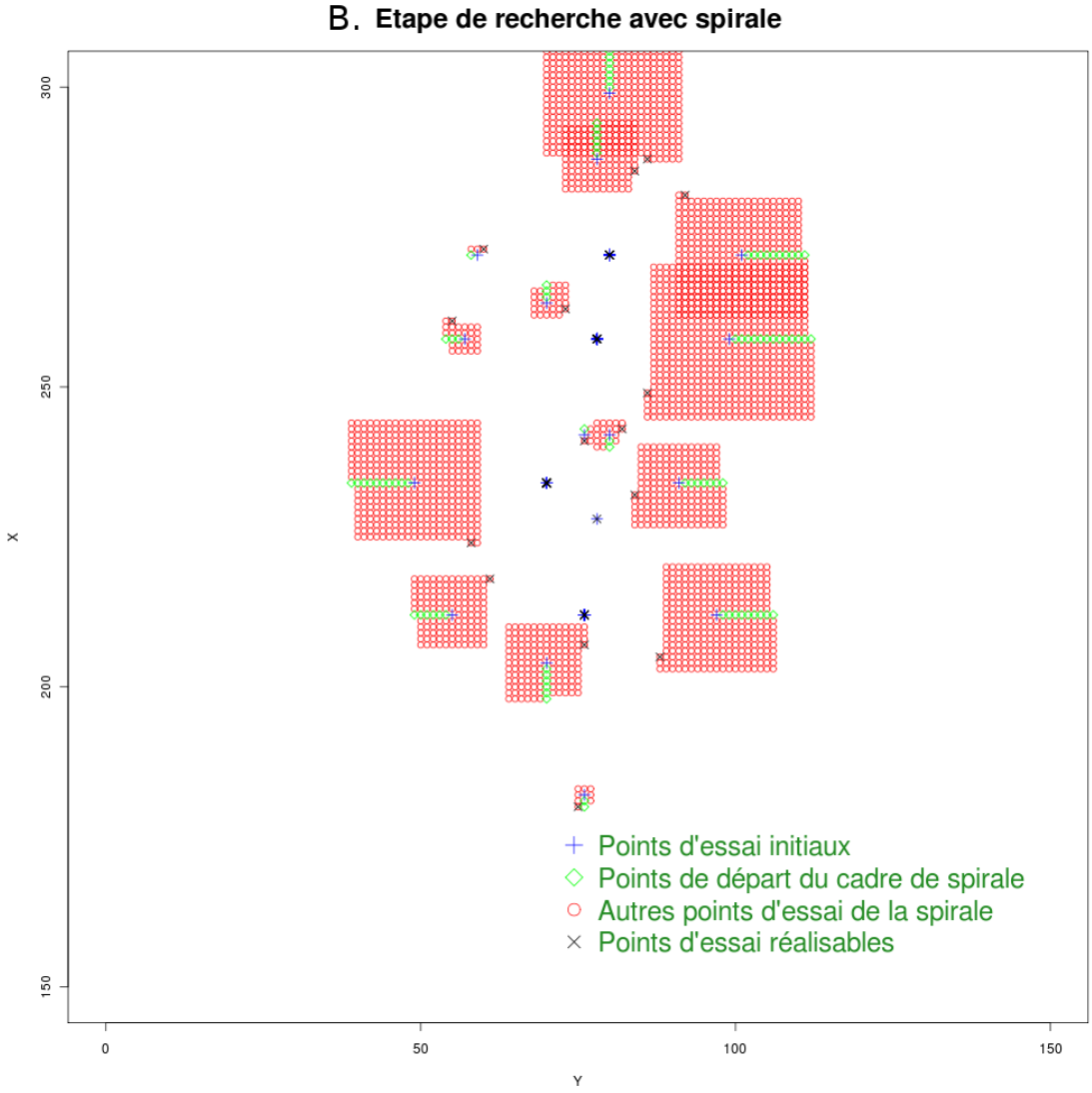

FIGURE 5.5 Une méthode de recherche en spirale pour 5 balises (donc 10 variables et 20 points d'essai), avec une demi-largeur de 25 pixels.

5.4 Résultats numériques avec la fonction substitut

L'ensemble des méthodes pouvant être obtenues par combinaisons entre les différentes possibilités précédemment décrites ont été testées (Figure 5.6) et vont être comparées dans cette section. Pour faciliter au lecteur l'assimilation des dénominations algorithmiques, ces dernières et leur signification ont été récapitulées à la page xx de la section *"Sigles, abréviations et notations"* précédant le chapitre 1.

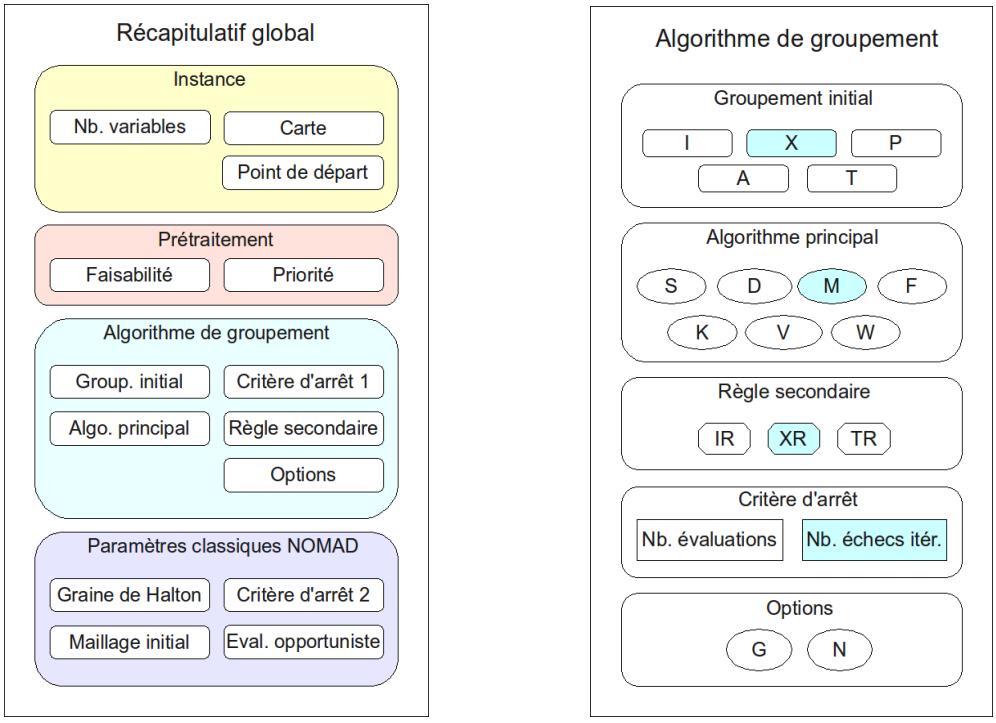

FIGURE 5.6 À gauche, une vision haut-niveau du banc d'essai. À droite, un exemple d'algorithme générique, intitulé "X_M_XR" (le critère d'arrêt EC n'est pas mentionné car pris par défaut) et basé sur une gestion dynamique par mouvement.

5.4.1 Méthodologie

Pour chaque stratégie, on a effectué des tests sur 18 instances. Il y a plus d'une centaine d'algorithmes à comparer si l'on se tient avec rigueur aux possibilités définies par le banc

d'essai. Cela représente donc une grande quantité de données à analyser, qu'il va être nécessaire de synthétiser en agrégeant l'information.

Rappelons que nous avions fixé pour les tests effectués un budget d'évaluation maximal à 1000 évaluations de boîte noire. Nous allons définir deux types de budgets d'évaluations : les budgets d'évaluations totaux (BET), indépendant de l'instance, et les budgets d'évaluations par GMON (BPG), fonction du nombre de GMON. Pour chacun de ces types, nous allons considérer trois budgets arbitraires : $BET \in \{250, 500, 1000\}$ et $BPG \in \{30, 70, 100\}$. Par exemple, pour une configuration à 8 GMON, un budget d'évaluations par GMON égal à 30 donnerait un budget total d'évaluations de $8 \times 30 = 240$, et ceci indépendamment de la carte considérée. Afin de comparer toutes nos stratégies, nous allons utiliser deux méthodes complémentaires : le calcul de moyennes basées sur les *améliorations relatives*, ainsi que les *profils de performance*, ces derniers ayant été originalement introduits par Dolan et Moré (2002). La première va nous permettre d'extraire les meilleurs algorithmes par l'estimation numérique des performances, puis la seconde nous aidera à visualiser ces dernières afin de conclure de manière pertinente. Rappelons tout d'abord comment les obtenir, afin d'être en mesure de les exploiter par la suite.

Amélioration relative

Supposons qu'un algorithme a, inclus dans un ensemble A d'algorithmes à comparer, renvoit une valeur objectif positive $v_p(a, b)$ lorsqu'il est éxécuté sur une instance p incluse dans un ensemble de problèmes P, avec un budget d'évaluations b inclus dans un ensemble de budgets B, à partir d'une solution initiale x_0^p de valeur $f(x_0^p) > 0$.

Le principe est simple : on va effectuer une moyenne des améliorations relatives sur l'ensemble des instances pour assigner un score $\eta(a, b)$ à un algorithme a pour un budget fixé à b. Ensuite, la moyenne sur l'ensemble des budgets $\bar{\eta}(a)$ servira à classer les algorithmes dans les tableaux pour en accroître la visibilité. Voici la méthode pour les calculer :

$$\eta(a, b) = \frac{1}{|P|} \sum_{p \in P} \frac{f(x_0^p) - v_p(a, b)}{f(x_0^p)} \quad ,$$

$$\bar{\eta}(a) = \frac{1}{|B|} \sum_{b \in B} \eta(a, b) \ .$$

Ces deux indices sont adimensionnels et positifs, dès lors que $f(x_0^p) \geq v_p(a, b), \forall a, b, p$ dans un contexte de minimisation : à budget b fixé, l'algorithme ayant le score $\eta(a, b)$ le plus élevé est le meilleur en moyenne sur l'ensemble des instances P. Notons que $\eta(a, b)$ est une fonction croissante de b. Ainsi, si les budgets sont suffisamment bien répartis, un algorithme a_1 atteignant dès les premières évaluations son optimum aura une valeur $\bar{\eta}(a_1) > \bar{\eta}(a_2)$, si a_2 atteint plus lentement un optimum de même valeur, ce qui est cohérent.

Profils de performance (Dolan et Moré (2002))

Supposons qu'un algorithme a, inclus dans un ensemble A d'algorithmes à comparer, renvoit une valeur objectif positive $v_p(a)$ lorsqu'il est éxécuté sur une instance p incluse dans un ensemble de problèmes P. Dans un contexte de minimisation, le meilleur algorithme pour l'instance $p \in P$ renvoit la valeur objectif la plus faible v_p^*. Définissons pour tout $v, v^* \geq 0$ et $\alpha \geq 1$, la fonction de score :

$$s(v, v^*, \alpha) = \begin{cases} 1 & \text{si } v \leq \alpha v^*, \\ 0 & \text{sinon.} \end{cases}$$

Le profil de performance d'un algorithme $a \in A$ est la fonction associant à une valeur α la moyenne des scores de l'algorithme :

$$\zeta_a(\alpha) = \frac{1}{|P|} \sum_{p \in P} s(v_p(a), v_p^*, \alpha) \ .$$

Remarquons d'emblée qu'en tant que moyenne de valeurs binaires, $0 \leq \zeta_a(\alpha) \leq 1$. De plus, pour tout $a \in A$, $\zeta_a(1)$ correspond au pourcentage d'instances où l'algorithme a est le meilleur, $\zeta_a(2)$ correspond au pourcentage d'instances où l'algorithme a est à un facteur 2 du meilleur, et $\lim_{\alpha \to \infty} \zeta_a(\alpha)$ correspond au pourcentage d'instances sur lequel l'algorithme a réussi. Ici, puisque nous partons d'une solution réalisable, cette limite sera toujours égale à 1. Enfin, notons que la fonction ζ_a est monotone croissante sur $[1, +\infty[$.

Organisation des résultats

Nous allons dans un premier temps analyser les performances des algorithmes statiques avec et sans prétraitement, puis nous choisirons en fonction des résultats de poursuivre avec la totalité ou une partie des prétraitements. Nous tenterons ensuite de nous ramener "brutalement" au cas classique, par l'ajout d'un algorithme MADS classique soit à la fin de l'algorithme, soit "en parallèle" (il ne s'agit pas de parallélisme informatique à proprement parler, mais simplement d'un abus de langage pour signifier l'ajout d'un groupe N contenant toutes les variables à chaque itération).

Enfin, nous passerons aux groupements dynamiques évolutifs. Les règles secondaires IR, XR et TR, destinées au groupement des variables n'étant pas directement gérées par l'algorithme de reconfiguration, sont explicitement mentionnées dans le nom de l'algorithme lorsqu'elles sont activées. Nous allons considérer toutes les combinaisons possibles en trois mouvements. Tout d'abord, nous allons nous intéresser aux algorithmes principaux qui ne groupent qu'un sous-ensemble des variables, et nécessitent l'emploi d'une règle secondaire. Nous allons débuter avec l'algorithme D, basé sur une fusion progressive par distance simple et intuitive. Puis, nous verrons les algorithmes basés sur l'absence de mouvement, à savoir M et sa variante F. Enfin, nous traiterons ensemble les algorithmes K, V et W, qui ne requièrent pas de règle secondaire et groupent l'ensemble des variables. Nous extrairons finalement les champions de chaque classe et les comparerons.

Rappelons par ailleurs que nous avons résumé la signification des différentes dénominations utilisées à la page xx (au début du mémoire).

5.4.2 Pré-traitement

Voici les résultats dans le cas statique pour différents budgets d'évaluation :

- Sans aucun prétraitement (Tableau 5.1).
- Avec prétraitement sur la réalisabilité (Tableau 5.2).
- Avec prétraitement sur la réalisabilité et sur l'ordre d'évaluation (Tableau 5.3).

On observe globalement trois résultats intéressants. Premièrement, prétraiter pour corriger la réalisabilité et ordonner les points à évaluer selon la valeur de l'erreur courante sont deux ajouts bénéfiques. Notons qu'en accord avec l'intuition, dès lors que nous traitons

TABLEAU 5.1 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **statique**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés, **sans aucun prétraitement**.

		BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	S	0.89	0.92	0.92	0.80	0.92	0.92	0.89
T	S	0.77	0.77	0.77	0.77	0.77	0.77	0.77
A	S	0.77	0.78	0.78	0.73	0.78	0.78	0.77
P	S	0.65	0.67	0.68	0.65	0.68	0.68	0.67
I	S	0.42	0.42	0.42	0.42	0.42	0.42	0.42

TABLEAU 5.2 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **statique**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés, avec **prétraitement** pour corriger la **réalisabilité** des points d'essai.

		BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	S	1.31	1.60	1.71	1.27	1.53	1.68	1.52
P	S	1.29	1.55	1.73	1.15	1.52	1.64	1.48
T	S	1.23	1.55	1.71	1.18	1.50	1.66	1.47
I	S	1.25	1.53	1.70	1.20	1.51	1.59	1.46
A	S	1.05	1.43	1.69	0.88	1.32	1.52	1.31

TABLEAU 5.3 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **statique**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés, avec **prétraitement** pour corriger la **réalisabilité** des points d'essai et **priorité d'évaluation** basée sur l'erreur courante.

		BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
I	S	1.40	1.65	1.74	1.34	1.61	1.71	1.57
X	S	1.36	1.61	1.70	1.29	1.59	1.67	1.54
P	S	1.31	1.54	1.71	1.28	1.55	1.61	1.50
A	S	1.35	1.55	1.65	1.24	1.55	1.63	1.49
T	S	1.30	1.54	1.64	1.25	1.53	1.63	1.48

avec des domaines fragmentés, il y a une nette différence entre le cas sans prétraitement et celui avec correction de la réalisabilité. Une exploration plus en détail des valeurs numériques montrent que pour plusieurs instances, le nombre d'évaluations effectuées a été très faible (notamment pour la carte GAT où le nombre d'évaluations effectuées, tout algorithme statique confondu, n'a jamais dépassé 31).

Deuxièmement, grouper les variables est utile : l'algorithme MADS classique (T_S) est ici très souvent dominé (deuxième dans le cas sans prétraitement, troisième avec prétraitement pour la faisabilité, dernier avec prétraitement pour la faisabilité et priorité d'évaluation). Mentionnons cependant que nous travaillons ici avec une fonction substitut qui est une représentation très simplifiée de la boîte noire originale : nous ne pouvons évidemment pas postuler que T_S sera systématiquement dominé et nous le conserverons comme base de comparaison par la suite.

Troisièmement, avec tous les prétraitements, les groupements basés sur des groupes de petite taille sont plus efficaces que les autres : en effet, les algorithmes apparaissent classés par ordre de taille. Le groupement initial I offre des performances irrégulières (dernier, avant-dernier, premier), tandis que l'algorithme X apparaît robuste (premier, premier, second). Les groupements les plus intuitifs (I et X) dominent les autres (P, A et T). Enfin, notons que la combinaison de l'algorithme I_S avec les prétraitements apparaît la stratégie dominante pour les algorithmes statiques.

Gardons à l'esprit que d'après la précédente section 4.3, les groupements statiques (à l'exception de T_S) ne disposent pas de propriétés de convergence aussi fortes que le cas classique, ce à quoi nous allons tenter de remédier.

5.4.3 Ajout d'un MADS classique

L'expression "ajouter un MADS classique" est un terme commode (qui n'a pas d'autre vocation que d'alléger le texte) pour signifier que l'on éxécute l'algorithme MADS classique à la suite d'une heuristique, à partir de la meilleure solution trouvée.

Par la suite, on considère que le prétraitement (réalisabilité et priorité d'évaluation) est bénéfique : il n'est plus remis en question et est appliqué par défaut à l'ensemble des algorithmes (sauf mention contraire).

À la fin

Nous procédons ici au rajout d'un MADS classique à partir d'un certain nombre d'éxécutions, la transition s'effectuant grâce à l'utilisation du critère d'arrêt EC précédemment évoqué basé sur les échecs consécutifs. L'algorithme termine par un MADS classique au bout d'un nombre de phases d'exécutions égal au nombre de GMON, avec prétraitement pour la réalisabilité et l'ordre d'évaluation (Tableau 5.4). On ajoute la lettre "G" aux dénominations des algorithmes pour permettre une distinction avec les précédents.

TABLEAU 5.4 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **statique**, pour un **nombre d'exécutions égal au nombre de GMON** (G), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG).

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
I	S	G	1.40	1.65	1.75	1.34	1.61	1.71	1.58
X	S	G	1.39	1.61	1.71	1.32	1.61	1.68	1.55
P	S	G	1.39	1.60	1.71	1.32	1.60	1.66	1.55
T	S	G	1.33	1.55	1.74	1.25	1.55	1.64	1.51
A	S	G	1.31	1.55	1.74	1.24	1.54	1.63	1.50

$\underline{S\_G}$: En ajoutant simplement un MADS classique après $g - 1$ exécutions (g étant le nombre de GMON), on obtient assez logiquement un classement peu différent du précédent Tableau 5.3, où les stratégies de groupements de faible taille (I en tête, puis X et P) dominent. Viennent ensuite les groupements T et A, de performances moyennes semblables. Bien qu'ayant des difficultés dans les premières centaines d'itérations, celles-ci affichent un score honorable pour le budget maximum. Notons que nous héritons ici des propriétés théoriques de convergence inhérentes au cas classique.

Cependant, les groupes de variables ne peuvent se déplacer que dans leur sous-espace associé jusqu'à ce qu'on déclenche le MADS classique, ce qui semble peu optimal. Cette relaxation est cependant bénéfique, puisqu'on constate que toutes les moyennes ont augmentées, la meilleure valeur atteinte étant désormais 1.75 (au lieu du précédent 1.74). La continuité entre deux exécutions n'est apparemment pas totale, dès lors que T_S et T_S_G n'ont pas eu des déroulements identiques.

En parallèle

Le groupe N contenant toutes les variables est ajouté à chaque itération de l'exécution. Étant donné que nous utilisons une approche opportuniste avec un ordre d'évaluation particulier, les points de sonde rajoutés peuvent permettre de converger vers une solution d'excellente qualité sans trop souffrir d'un excès d'évaluations. L'inconvénient est que, dans certains cas (par exemple, si l'ordre d'évaluation est peu pertinent), ce surplus d'évaluations sera net et coûteux par rapport au cas sans ajout. On ajoute la lettre "N" aux dénominations des algorithmes pour rendre aisée la distinction avec les précédents résultats. On peut observer les résultats statiques avec N au Tableau 5.5.

TABLEAU 5.5 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **statique**, avec **groupe N** (N), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	S	N	1.28	1.54	1.75	1.21	1.54	1.65	1.49
P	S	N	1.29	1.55	1.74	1.22	1.52	1.64	1.49
T	S	N	1.30	1.54	1.64	1.25	1.53	1.63	1.48
A	S	N	1.28	1.54	1.66	1.20	1.53	1.63	1.47
I	S	N	1.26	1.53	1.70	1.18	1.49	1.63	1.46

$\underline{S\_N}$: Le groupement X domine avec P, tandis que I est relégué à la dernière place, montrant une grande irrégularité. On constate que la moyenne pour les budgets considérés est faible comparée au cas sans ajout N, car le déroulement est lent. Cependant, la solution pour un budget de 1000 évaluations est parfois d'excellente qualité. En effet, l'algorithme ayant énormément de possibilités de directions, il va converger en général vers un minimum local de qualité, au détriment du nombre d'évaluations. Logiquement, on observe que les cas T_S et T_S_N sont identiques puisque les groupes "doublons" sont effacés par NOMAD avant l'exécution.

À la fin et en parallèle

Le groupe N est ajouté à chaque itération de l'algorithme statique durant $g - 1$ exécutions, puis on lance un algorithme MADS classique. On peut observer les résultats

numériques pour les algorithmes statiques avec g exécutions finissant par un algorithme MADS classique au Tableau 5.6).

TABLEAU 5.6 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **statique**, pour un **nombre d'exécutions égal au nombre de GMON** (G), avec **groupe N** (N), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

				BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	S	G	N	1.30	1.59	1.76	1.26	1.54	1.66	1.52
P	S	G	N	1.32	1.57	1.76	1.21	1.56	1.70	1.52
T	S	G	N	1.33	1.55	1.74	1.25	1.55	1.64	1.51
A	S	G	N	1.30	1.54	1.69	1.20	1.53	1.64	1.49
I	S	G	N	1.26	1.53	1.71	1.18	1.49	1.63	1.47

S_G_N : Le classement est identique au cas S_N, mais les valeurs numériques ont évolué. La performance moyenne a augmenté, ainsi que la qualité des solutions pour le budget maximal qui plafonne désormais à 1.76.

5.4.4 Avec distance (D)

Rappelons que le critère d'arrêt pour reconfiguration retenu (pour passer d'une phase d'exécution à l'autre) est EC. On peut observer les résultats numériques dans le cas dynamique pour différents budgets d'évaluations :

– Avec distance D (Tableau 5.7).

– Avec distance D_N (Tableau 5.8).

D : Globalement, le groupement initial X apparaît clairement dominant, tandis que I affiche des performances irrégulières, A et P sont moyens, et T est dominé. La règle secondaire XR est bonne, la règle IR, bien qu'ayant permis l'obtention de la meilleure stratégie, est irrégulière, et toutes deux dominent TR.

D_N : Les règles externes apparaissent par taille décroissante : TR domine globalement XR, et toutes deux dominent largement la règle IR. Les groupements initiaux X et P semblent dominer T, A et finalement I.

TABLEAU 5.7 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **dynamique**, basé sur la **distance**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	D	IR	1.36	1.61	1.76	1.30	1.57	1.69	1.55
I	D	XR	1.38	1.59	1.72	1.28	1.60	1.69	1.54
P	D	TR	1.33	1.58	1.75	1.28	1.58	1.66	1.53
A	D	IR	1.35	1.59	1.73	1.29	1.56	1.66	1.53
X	D	XR	1.33	1.60	1.71	1.27	1.60	1.68	1.53
P	D	XR	1.32	1.58	1.76	1.26	1.56	1.67	1.52
X	D	TR	1.35	1.58	1.71	1.26	1.58	1.67	1.52
A	D	XR	1.36	1.58	1.68	1.27	1.60	1.65	1.52
T	D	XR	1.30	1.59	1.73	1.26	1.56	1.67	1.52
I	D	IR	1.36	1.58	1.69	1.28	1.55	1.65	1.52
P	D	IR	1.35	1.58	1.69	1.27	1.56	1.65	1.52
A	D	TR	1.33	1.57	1.69	1.25	1.56	1.67	1.51
T	D	IR	1.30	1.59	1.72	1.24	1.56	1.65	1.51
T	D	TR	1.30	1.57	1.74	1.25	1.53	1.65	1.51
I	D	TR	1.29	1.55	1.76	1.24	1.50	1.64	1.50

TABLEAU 5.8 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **dynamique**, basé sur la **distance**, avec **groupe N** (N), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

				BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
P	D	TR	N	1.29	1.59	1.75	1.25	1.56	1.67	1.52
X	D	TR	N	1.34	1.56	1.72	1.26	1.56	1.66	1.52
P	D	XR	N	1.31	1.57	1.75	1.23	1.52	1.66	1.51
T	D	TR	N	1.33	1.56	1.68	1.25	1.55	1.64	1.50
A	D	TR	N	1.29	1.57	1.68	1.23	1.56	1.65	1.50
A	D	XR	N	1.29	1.55	1.71	1.22	1.54	1.64	1.49
I	D	TR	N	1.30	1.55	1.68	1.21	1.56	1.65	1.49
P	D	IR	N	1.31	1.53	1.75	1.19	1.52	1.64	1.49
X	D	XR	N	1.26	1.56	1.70	1.23	1.54	1.65	1.49
T	D	XR	N	1.29	1.54	1.70	1.21	1.54	1.64	1.49
I	D	XR	N	1.29	1.52	1.71	1.22	1.51	1.62	1.48
X	D	IR	N	1.28	1.53	1.70	1.19	1.52	1.62	1.47
T	D	IR	N	1.26	1.52	1.70	1.20	1.51	1.63	1.47
I	D	IR	N	1.25	1.50	1.68	1.17	1.49	1.63	1.45
A	D	IR	N	1.27	1.50	1.68	1.16	1.50	1.62	1.45

5.4.5 Avec mouvement (M et F)

On peut observer les résultats numériques dans le cas dynamique pour différents budgets d'évaluations :

- Avec mouvement M et F (Tableau 5.9).
- Avec mouvement M_N et F_N (Tableau 5.10).

TABLEAU 5.9 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **dynamique** basé sur le **mouvement**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
I	M	IR	1.40	1.60	1.75	1.28	1.59	1.67	1.55
I	F	IR	1.37	1.58	1.74	1.29	1.58	1.68	1.54
X	M	IR	1.38	1.60	1.70	1.25	1.59	1.68	1.53
P	F	IR	1.39	1.57	1.71	1.29	1.57	1.67	1.53
P	M	IR	1.39	1.56	1.70	1.29	1.56	1.66	1.53
P	F	XR	1.36	1.58	1.68	1.30	1.59	1.65	1.53
P	M	XR	1.36	1.57	1.68	1.29	1.58	1.65	1.52
X	M	TR	1.35	1.58	1.72	1.26	1.57	1.65	1.52
X	F	XR	1.35	1.57	1.71	1.27	1.56	1.66	1.52
A	M	XR	1.35	1.57	1.69	1.27	1.58	1.66	1.52
A	M	IR	1.34	1.58	1.69	1.23	1.58	1.67	1.51
X	F	TR	1.34	1.57	1.71	1.26	1.56	1.65	1.51
I	M	XR	1.35	1.57	1.68	1.26	1.56	1.66	1.51
I	F	TR	1.34	1.56	1.73	1.25	1.55	1.64	1.51
I	M	TR	1.34	1.55	1.73	1.25	1.55	1.64	1.51
I	F	XR	1.35	1.56	1.69	1.25	1.55	1.65	1.51
A	F	XR	1.35	1.56	1.67	1.26	1.57	1.64	1.51
T	M	XR	1.34	1.56	1.69	1.25	1.54	1.65	1.51
A	F	IR	1.32	1.56	1.69	1.23	1.56	1.66	1.50
X	M	XR	1.30	1.55	1.72	1.27	1.52	1.63	1.50
T	F	IR	1.34	1.55	1.67	1.23	1.57	1.65	1.50
X	F	IR	1.33	1.55	1.70	1.25	1.53	1.62	1.50
P	F	TR	1.32	1.54	1.68	1.29	1.53	1.61	1.50
P	M	TR	1.32	1.54	1.68	1.28	1.53	1.62	1.50
T	M	TR	1.29	1.53	1.75	1.26	1.51	1.61	1.49
T	M	IR	1.30	1.54	1.71	1.23	1.54	1.63	1.49
A	F	TR	1.31	1.55	1.66	1.23	1.55	1.64	1.49
T	F	TR	1.30	1.54	1.65	1.25	1.52	1.63	1.48
T	F	XR	1.30	1.54	1.65	1.25	1.51	1.62	1.48
A	M	TR	1.28	1.53	1.67	1.23	1.52	1.62	1.48

$\underline{M \text{ et } F}$: La règle secondaire IR apparaît dominante, suivie de XR, puis de TR. Globalement, les groupements initiaux en ordre de performance décroissante sont I, puis P et X, et enfin A et T. Notons que l'on ne peut pas extraire de domination claire entre les algorithmes de groupements principaux M et F, et ces algorithmes affichent des valeurs de fonction objectif stables et relativement peu dispersées.

TABLEAU 5.10 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **dynamique** basé sur le **mouvement**, avec **groupe N** (N), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

				BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
P	M	XR	N	1.36	1.58	1.76	1.24	1.57	1.69	1.53
P	F	IR	N	1.36	1.57	1.74	1.25	1.57	1.68	1.53
X	M	XR	N	1.37	1.57	1.72	1.26	1.57	1.66	1.53
P	M	TR	N	1.32	1.57	1.75	1.24	1.54	1.67	1.52
P	F	TR	N	1.33	1.56	1.69	1.24	1.55	1.66	1.50
A	M	XR	N	1.32	1.56	1.70	1.24	1.54	1.65	1.50
T	M	TR	N	1.33	1.55	1.68	1.24	1.54	1.63	1.50
A	M	TR	N	1.30	1.56	1.72	1.24	1.51	1.65	1.50
T	F	XR	N	1.33	1.55	1.66	1.25	1.55	1.64	1.50
T	M	XR	N	1.28	1.56	1.69	1.25	1.54	1.65	1.49
X	F	IR	N	1.31	1.54	1.67	1.27	1.53	1.64	1.49
A	F	TR	N	1.32	1.56	1.67	1.24	1.53	1.65	1.49
T	F	IR	N	1.30	1.54	1.68	1.22	1.53	1.66	1.49
A	F	IR	N	1.28	1.54	1.71	1.23	1.53	1.65	1.49
X	M	TR	N	1.30	1.54	1.70	1.26	1.50	1.63	1.49
I	F	TR	N	1.29	1.54	1.70	1.22	1.53	1.63	1.49
X	F	XR	N	1.31	1.54	1.69	1.25	1.52	1.61	1.49
X	F	TR	N	1.31	1.52	1.68	1.26	1.51	1.61	1.48
X	M	IR	N	1.28	1.53	1.67	1.26	1.52	1.63	1.48
I	M	TR	N	1.30	1.54	1.70	1.21	1.53	1.62	1.48
P	F	XR	N	1.32	1.51	1.65	1.25	1.54	1.62	1.48
A	F	XR	N	1.29	1.54	1.68	1.24	1.51	1.63	1.48
T	F	TR	N	1.29	1.54	1.66	1.24	1.51	1.62	1.48
I	F	XR	N	1.29	1.52	1.68	1.20	1.51	1.64	1.47
I	M	XR	N	1.30	1.52	1.67	1.21	1.51	1.64	1.47
T	M	IR	N	1.25	1.52	1.68	1.22	1.51	1.63	1.47
I	F	IR	N	1.27	1.50	1.72	1.20	1.51	1.61	1.47
A	M	IR	N	1.25	1.51	1.68	1.23	1.50	1.63	1.47
I	M	IR	N	1.27	1.50	1.72	1.20	1.50	1.60	1.46
P	M	IR	N	1.28	1.50	1.67	1.25	1.48	1.60	1.46

$\underline{M\_N \text{ et } F\_N :}$ La règle secondaire XR, puis TR dominent clairement IR (si l'on excepte l'excellente performance de $P\_F\_IR\_N$). Sept des dix stratégies ayant les meilleures moyennes sont obtenues avec M, et les cinq premières utilisent des groupements initiaux par paires (P et X). Les groupements initiaux P dominent (bien que P soit également dans la pire stratégie), X, T, A apparaissent dans le peloton et I semblent globalement le moins bon.

5.4.6 Avec classification et sensibilité (K, V et W)

On peut observer les résultats numériques dans le cas dynamique pour différents budgets d'évaluations :

- Avec classification K, et sensibilité V et W (Tableau 5.11).
- Avec classification K_N, et sensibilité V_N et W_N (Tableau 5.12).

TABLEAU 5.11 Amélioration relative (valeur absolue en %) de la fonction objectif pour chaque algorithme **dynamique** basé sur la **classification** et la **sensibilité**, pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

		BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
X	V	1.44	1.66	1.76	1.36	1.65	1.72	1.60
A	V	1.42	1.63	1.77	1.35	1.59	1.72	1.58
T	V	1.40	1.62	1.77	1.34	1.60	1.70	1.57
I	V	1.42	1.63	1.70	1.34	1.60	1.68	1.56
P	V	1.35	1.60	1.76	1.30	1.58	1.68	1.55
P	K	1.32	1.59	1.76	1.27	1.58	1.67	1.53
P	W	1.34	1.60	1.74	1.27	1.55	1.66	1.53
X	K	1.34	1.60	1.70	1.28	1.58	1.67	1.53
X	W	1.37	1.56	1.74	1.29	1.54	1.65	1.53
A	K	1.34	1.60	1.70	1.26	1.59	1.66	1.52
I	K	1.33	1.59	1.70	1.27	1.58	1.67	1.52
T	W	1.33	1.57	1.75	1.28	1.55	1.66	1.52
I	W	1.35	1.59	1.69	1.28	1.57	1.65	1.52
A	W	1.33	1.58	1.72	1.26	1.56	1.65	1.52
T	K	1.29	1.56	1.75	1.27	1.50	1.65	1.50

$\underline{K, V \text{ et } W :}$ À groupement initial fixé, l'algorithme de reconfiguration V apparaît clairement comme la meilleure stratégie, suivi de K puis de W. À algorithme de reconfiguration

fixé, les groupements initiaux par paires X et P apparaissent nettement dans les meilleurs à une irrégularité près (premier, second, second pour X et dernier, premier, premier pour P). Notons que X_V est dominant sur l'ensemble des stratégies considérées jusqu'ici. Enfin, mentionnons que A_V et T_V possèdent une valeur objectif finale dominante pour le budget maximum.

TABLEAU 5.12 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **dynamique** basé sur la **classification** et la **sensibilité**, avec **groupe N** (N), pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés.

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
P	K	N	1.31	1.57	1.74	1.22	1.52	1.66	1.50
P	W	N	1.33	1.55	1.72	1.22	1.53	1.65	1.50
A	W	N	1.28	1.55	1.73	1.23	1.53	1.65	1.50
X	V	N	1.25	1.57	1.71	1.20	1.57	1.65	1.49
X	W	N	1.29	1.55	1.68	1.23	1.55	1.65	1.49
P	V	N	1.31	1.53	1.72	1.22	1.49	1.63	1.48
T	V	N	1.28	1.54	1.71	1.22	1.53	1.63	1.48
A	V	N	1.28	1.55	1.69	1.20	1.53	1.64	1.48
T	K	N	1.27	1.55	1.73	1.21	1.51	1.63	1.48
T	W	N	1.28	1.53	1.69	1.24	1.51	1.62	1.48
I	V	N	1.25	1.56	1.69	1.18	1.54	1.65	1.48
X	K	N	1.28	1.54	1.68	1.23	1.50	1.63	1.48
A	K	N	1.25	1.54	1.70	1.21	1.52	1.63	1.47
I	K	N	1.29	1.51	1.68	1.22	1.51	1.62	1.47
I	W	N	1.25	1.53	1.67	1.17	1.52	1.61	1.46

K_N, V_N et W_N : Les algorithmes principaux K et W sont très irréguliers (dans les premiers et derniers), tandis que V, s'il ne fait pas partie du trio de tête, offre des performances bonnes et stables. Les groupements initiaux par paires (P puis X) dominent les autres, I et T se partageant les pires scores.

Finalement, on peut observer les profils de performance pour l'ensemble des algorithmes dynamiques sur la Figure 5.7. Cette dernière, d'apparence très chargée, n'est pas destinée à exposer clairement chacune des performances algorithmiques, mais simplement à illustrer deux faits notables : l'algorithme X_V se distingue nettement tandis que T_S est globalement dominé par les stratégies dynamiques.

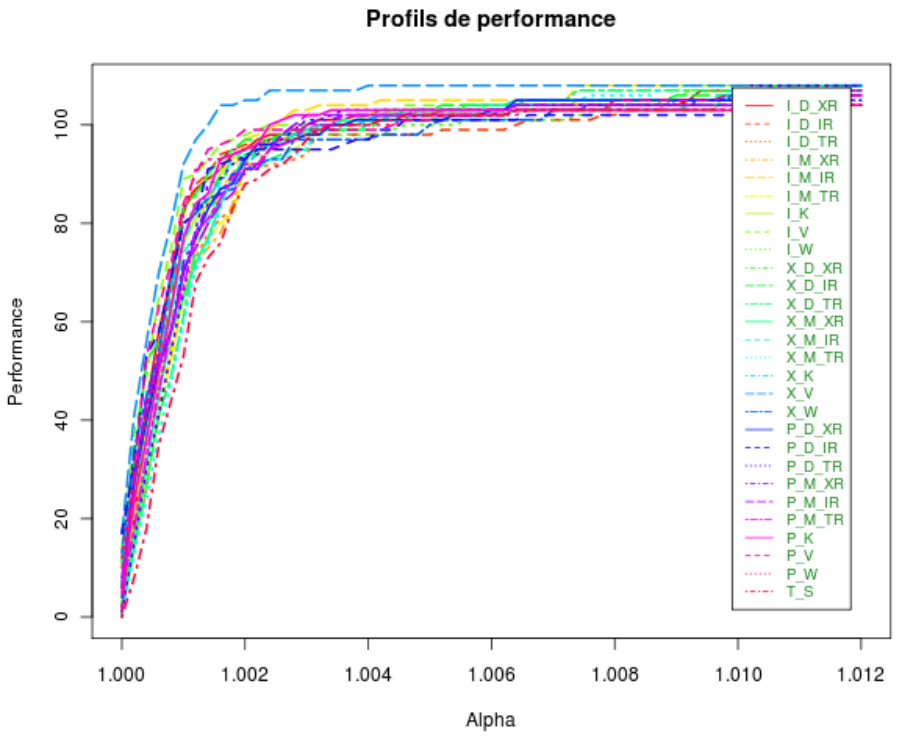

FIGURE 5.7 Profils de performances pour chaque algorithme **dynamique** (les N, T et A, peu performants, ont été retirés pour ne pas abusivement détériorer la lisibilité) sur 108 problèmes (18 instances $\times$ 6 budgets).

5.4.7 Stratégies retenus pour l'IREQ

L'IREQ nous accorde des ressources informatiques pour appliquer huit stratégies algorithmiques sur le problème original. Nous cherchons donc à effectuer un tri parmi l'ensemble des méthodes testées.

En premier lieu, nous délaissons les stratégies avec groupe N pour leurs performances peu convaincantes et leur gourmandise en évaluations, ainsi que les méthodes statiques terminant par une exécution de MADS classique (G) pour leur faible rapport gain/complexité.

Bien que l'algorithme de reconfiguration V ait livré d'excellentes performances numériques, nous ne souhaitons pas nous limiter à ses déclinaisons et préférons varier davantage les stratégies : nous sélectionnons donc, pour chaque type d'algorithme de reconfiguration dynamique, le meilleur candidat en moyenne. En tant qu'éléments de référence pour nos comparaisons, les algorithmes statiques I_S et T_S sont ajoutés à la sélection (Tableau 5.13 et Figure 5.8).

TABLEAU 5.13 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **sélectionné** pour plusieurs budgets d'évaluations totaux (BET) ou par GMON (BPG) donnés (cf. Figure 5.8).

			BET = 250	BET = 500	BET = 1000	BPG = 30	BPG = 70	BPG = 100	Moy.
Dynamiques									
X	V		1.44	1.66	1.76	1.36	1.65	1.72	1.60
X	D	IR	1.36	1.61	1.76	1.30	1.57	1.69	1.55
I	M	IR	1.40	1.60	1.75	1.28	1.59	1.67	1.55
I	F	IR	1.37	1.58	1.74	1.29	1.58	1.68	1.54
P	K		1.32	1.59	1.76	1.27	1.58	1.67	1.53
P	W		1.34	1.60	1.74	1.27	1.55	1.66	1.53
Statiques									
I	S		1.40	1.65	1.74	1.34	1.61	1.71	1.57
T	S		1.30	1.54	1.64	1.25	1.53	1.63	1.48

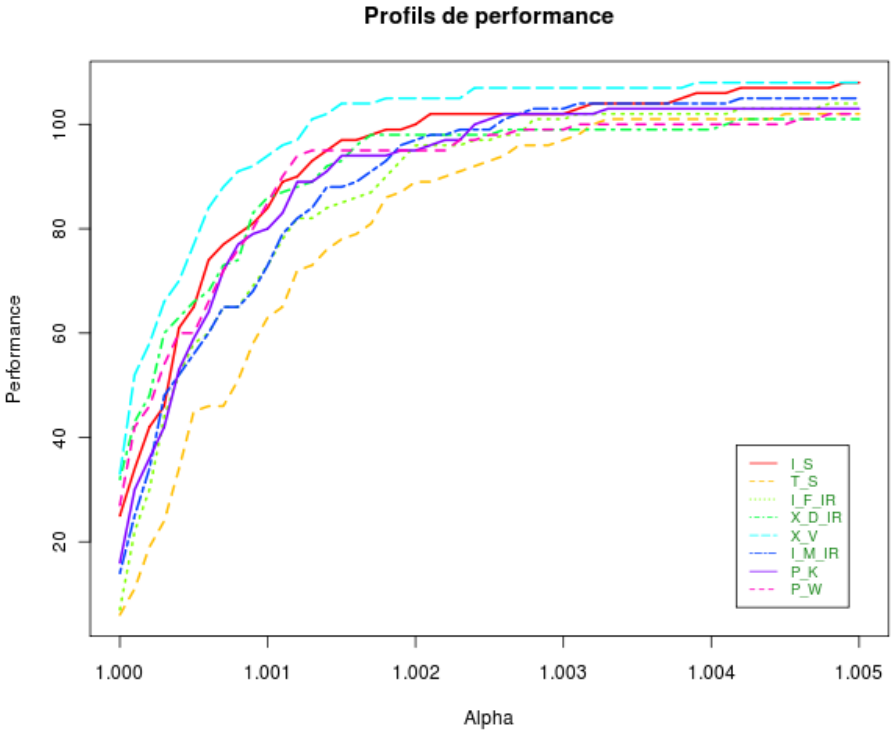

FIGURE 5.8 Profils de performances pour chaque algorithme **sélectionné** sur 108 problèmes (18 instances $\times$ 6 budgets) (cf. Tableau 5.13).

Chapitre 6

RÉSOLUTION DU PROBLÈME DE LOCALISATION DE BALISES GMON POUR LA MESURE DE L'ÉQUIVALENT EAU-NEIGE

Nous en venons à présent à la résolution pratique du problème original de positionnement de balises GMON sur les cartes de l'IREQ. Notre travail a jusqu'ici consisté à comparer et filtrer méthodiquement plus d'une centaine de variantes d'algorithmes dynamiques et statiques, ce qui a permis de distinguer six algorithmes de groupements dynamiques efficaces pour le problème substitut. Nous allons désormais comparer sur le problème original les performances des témoins statiques T_S (MADS classique) et I_S (Recherche par Coordonnées) aux meilleurs algorithmes dynamiques basés sur le groupement de variables, afin de démontrer ou réfuter l'utilité des groupes de variables sur ce type de problème.

6.1 La boîte noire de l'IREQ

Au moment de l'écriture de ce mémoire, l'aboutissement de la conception de la boîte noire n'est pas prévu avant plusieurs mois. De nombreuses idées sont en cours de développement : prises en compte des coûts d'implantation, éventuellement fonction de la distance au réseau routier ; coefficient de pondération des erreurs par sous-bassin, dès lors qu'il existe des zones où l'erreur est peu influente sur les processus décisionnels ; prise en compte de l'altitude et de la longitude pour l'interpolation géostatistique ; fusion de données entre

plusieurs dates réparties sur l'année pour obtenir des valeurs n'étant pas biaisées par des évènements ponctuels.

Cependant, des modèles de coût, de pondération et de fusion reposant sur des estimations expertes sont absents, et peu de ces initiatives sont actuellement exploitables. L'IREQ dispose d'une unique licence pour le logiciel ISATIS, essentiel aux calculs des modules de krigeages (et donc à l'obtention de la carte d'erreur), ce qui empêche la mise en parallèle des processus. De plus, seule la carte associée au bassin Saint-Maurice (STM) est pour l'instant utilisable avec ISATIS, ce qui nous oblige à restreindre notre plan d'expérience à celle-ci.

Mentionnons finalement que l'évaluation d'un point d'essai avec la boîte noire courante prend entre une et deux secondes. Le processus d'optimisation avec l'approche "boîte noire" est synthétisé au graphique 6.1.

6.2 Résultats numériques

6.2.1 Méthodologie

Le temps maximal imparti pour effectuer les tests à l'IREQ (comprenant essais préliminaires, mise en place des tests et temps de calcul) est d'une semaine. Nous allons travailler avec la seule carte disponible (STM), en reprenant pour celle-ci le domaine et les points de départ utilisés au chapitre 5. Ainsi, nous répétons exactement la même paramétrisation et démarche que pour le problème substitut, à ceci près que l'appel pour le calcul de la valeur objectif se fait dorénavant à la boîte noire de l'IREQ.

Concernant la priorité d'évaluation dans une optique opportuniste (où tout succès termine immédiatement l'itération courante), nous allons exploiter la fonction objectif du chapitre 5 : cette dernière jouera désormais le rôle de fonction substitut pour notre problème réel. Ainsi, à chaque itération, la liste des points d'essai est au préalable évaluée via la fonction substitut, puis les points sont réordonnés par valeur de fonction objectif substitut croissante (dans un contexte de minimisation), et enfin l'ensemble des points est évalué dans cet ordre avec la vraie fonction.

Nous avons été tributaires des aléas de l'optimisation pratique dans le monde industriel

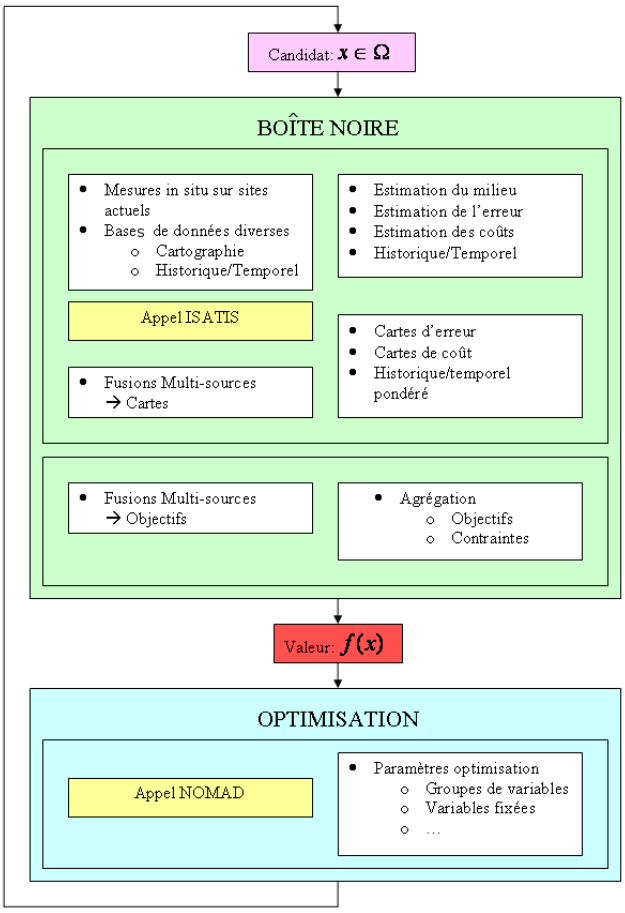

FIGURE 6.1 Processus d'optimisation sur le problème des GMON.

réel, qui sont apparus sous la forme d'erreurs de conception, de pannes, de paniques de noyaux et de limitations propres à la disponibilité des ressources (personnel, licences de logiciels, processeurs, cartes et temps). Ces imprévus ont freiné la mise en place des tests durant une période non négligeable de la visite à l'IREQ.

6.2.2 Optimisation de la fonction réelle avec la fonction substitut

Il est possible que la fonction substitut ne soit pas un indicateur de qualité pour l'ordre d'évaluation, auquel cas le nombre d'évaluations risque d'augmenter significativement, le classement des algorithmes reflètant principalement leur robustesse. En effet, nous avons vu au chapitre 5 que la priorité d'évaluation peut crucialement modifier les performances d'un algorithme avec des groupes statiques et il n'y a aucune raison de penser que ce n'est pas également le cas avec les groupements dynamiques.

Nous testons les 8 algorithmes sélectionnés sur les 6 instances disponibles avec un budget de 1000 évaluations, en utilisant la fonction substitut pour ordonner les points à évaluer. Cela devrait prendre selon nos estimations environ 1.33 jours de calcul.

Étant donné que seule la solution finale présente un intérêt dans le cadre du projet réel, nous nous concentrons uniquement sur la valeur objectif obtenue pour le budget maximal de 1000 itérations (Tableau 6.1).

TABLEAU 6.1 Amélioration relative (valeur absolue en $\%$) de la fonction objectif pour chaque algorithme **sélectionné**, avec **priorité d'évaluation basée sur la fonction substitut**, pour le budget maximal de 1000 itérations, avec un nombre de GMON g variant de 5 à 10.

			$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$	$g = 10$	Moy.
P	K		1.35	1.23	1.13	1.20	1.09	1.52	1.25
I	M	IR	1.33	1.27	1.13	1.16	1.10	1.52	1.25
P	W		1.34	1.24	1.14	1.16	1.08	1.51	1.25
X	D	IR	1.33	1.18	1.10	1.19	1.08	1.50	1.23
I	F	IR	1.31	1.27	0.97	1.17	1.10	1.51	1.22
T	S		1.31	1.22	0.99	1.16	1.08	1.51	1.21
X	V		1.32	1.27	1.06	1.12	0.91	1.23	1.15
I	S		1.02	1.17	1.12	1.11	0.91	1.22	1.09

Les algorithmes statiques (S) apparaissent assez clairement dominés en moyenne : les trois algorithmes ayant obtenu le meilleur score sont dynamiques, cinq des six algorithmes dynamiques ont devancé le MADS classique (T_S), et tous ont livré une meilleure performance que l'algorithme I_S, qui se retrouve désormais dernier. De manière surprenante, l'algorithme X_V, qui était largement dominant sur le modèle substitut, apparaît en bas du classement (juste devant I_S), dévoilant une grande irrégularité. Notons finalement que, pour chaque nombre de GMON considéré, le ou les meilleurs algorithmes sont systématiquement dynamiques (cf. cellules grisées du Tableau 6.1).

6.3 Synthèse

Depuis la section 5.4, nous avons entrepris de comparer un ensemble de stratégies plus ou moins pertinentes, exploitant ou non la structure du problème.

La taille des groupes Il apparaît que les groupements initiaux par paires (P et X) semblent les plus adéquats (X domine sur le problème substitut, tandis que P domine sur celui réel). En outre, les algorithmes gérant les variables avec des groupes initiaux ne contenant qu'une variable (I) et ceux avec groupement principal V, W ou avec règle secondaire IR semblent globalement assez irréguliers (ils amènent d'excellentes stratégies sur les modèles substituts et réels, mais également les pires dans le cas réel), tandis que ceux utilisant des groupes de grandes tailles (i.e., avec groupement initial T ou A, ou avec règle secondaire TR) n'offrent pas de bonnes performances sur le problème substitut. Ayant été contraint de limiter l'ensemble des tests, nous ne pouvons conclure avec certitude quant au mauvais comportement du groupement initial A ou de la règle TR sur le problème réel, d'autant que l'algorithme T_S obtient des résultats honorables (mais dominés) sur la vraie boîte noire.

Les algorithmes de reconfiguration On peut difficilement exhiber une relation d'ordre claire entre les algorithmes de reconfiguration, mais il semblerait qu'en moyenne, les algorithmes les plus prometteurs pour ce type de problèmes (réels) soient K, M et W. Remarquons que les algorithmes utilisant la notion de mémoire (D et F) ne livrent en

général pas de meilleures performances que leur homologue (K et M), ce qui tend à montrer qu'il faut fournir à l'algorithme la capacité d'"oublier" les évènements passés au cours de son déroulement. La médiocre performance de V peut s'expliquer par la forte non-linéarité du problème réel. En effet, il est probable que ce soit le caractère relativement lisse du problème substitut, qui ait permis à la régression linéaire "simple" de s'illustrer. Sur le problème original, la version W avec transformation des rangs, plus robuste, s'est alors distinguée.

Par ailleurs, contrairement à l'intuition, le groupement initial peut influencer grandement le déroulement de la résolution à algorithme de reconfiguration fixé. Ce fait peut être en partie attribué au critère d'arrêt EC, dont la première validation peut nécessiter plusieurs centaines d'évaluations.

Exploiter la structure de positionnement À l'inverse de ce qu'on pouvait attendre, nous n'avons pas retrouvé de domination distincte entre les algorithmes exploitant la structure du problème et les autres. En effet, le groupement initial P et les méthodes basées sur des régressions linéaires (V et W) ne l'exploitent pas directement. Toutefois, notons que deux des trois méthodes les plus performantes sur le problème original de l'IREQ (K et M) considèrent toujours les variables par paires relatives aux coordonnées des GMON et tirent donc profit des variables de positionnement.

Le potentiel des algorithmes dynamiques Si l'on fait abstraction de la performance de I_S sur le problème substitut (qui s'est finalement révélée être la pire stratégie parmi celles testées sur la vraie boîte noire), les algorithmes statiques (S ou S_N) ont toujours été clairement dominés par ceux dynamiques. En observant les valeurs des améliorations relatives pour les différents budgets, on constate que les groupes permettent (généralement) de progresser plus vite, tout en obtenant une solution aussi bonne, si ce n'est meilleure, qu'avec l'algorithme MADS classique. Plus précisément, cinq des six algorithmes dynamiques testés sur l'instance originale de l'IREQ ont devancé en moyenne le MADS classique (pour un budget de 1000 évaluations), et, pour chaque nombre de GMON testé, la meilleure solution obtenue a toujours été produite par un ou des algorithmes dynamiques.

Mise en perspective Par ailleurs, les résultats obtenus sur la boîte noire de l'IREQ, en cours de conception, sont à nuancer. En effet, l'indisponibilité des cartes GAT et LG pour ISATIS a limité nos tests à la seule carte STM, ce qui a réduit de deux tiers le plan d'expériences que nous avions développé au chapitre 5 avec le modèle substitut. De plus, la boîte noire originale ne prend en compte à présent que peu de paramètres, ce qui la rend finalement relativement aisée à résoudre : les valeurs objectif sont très peu dispersées, ce qui rend difficile les distinctions entre les performances algorithmiques. Certes, nous avons établi des moyennes sur plusieurs instances, avec différents budgets, pour tenter de rendre significatives les différences mineures, mais ils restent que ces différences sont intrinsèquement faibles.

Observons toutefois que les stratégies, bien que fournissant des valeurs d'améliorations relatives moyennes très semblables, offrent parfois des configurations significativement différentes au vu des dimensions du problème (Figure 6.2 avec 5, 7 et 9 GMON, Figure 6.3 avec 8 GMON). Notons également que les différences minimes entre les solutions obtenues par P_K, I_M_IR, P_W et T_S avec 10 GMON montrent que la diversité des solutions produites n'augmente pas forcément (et tend même à diminuer) avec le nombre de GMON.

La qualité du subsitut Nous pouvons voir sur la Figure 6.4 la différence entre les solutions obtenues avec notre fonction substitut et avec la boîte noire originale de l'IREQ, pour l'algorithme T_S avec un budget de 1000 évaluations. Les solutions apparaissent nettement distinctes, ce qui laissent supposer que la fonction subsitut n'était pas une réprésentation (simplifiée) fidèle à l'orignale. Elle nous a néanmoins permis de déterminer une solution d'excellente qualité pour le budget considéré.

Rappelons finalement que les stratégies dynamiques testées n'ont pas été spécialement optimisées pour obtenir de bonnes performances : dans l'optique de défricher le terrain des groupes de variables et de détecter un potentiel, nous avons décidé de couvrir plusieurs tailles de groupes, ces derniers étant gérés selon différentes logiques (par mouvement, distance ou sensibilité), avec un critère de reconfiguration particulièrement arbitraire, sans véritablement chercher à obtenir une unique technique optimale. Cela laisse à penser que ces méthodes dynamiques, déjà prometteuses, peuvent encore être largement améliorées.

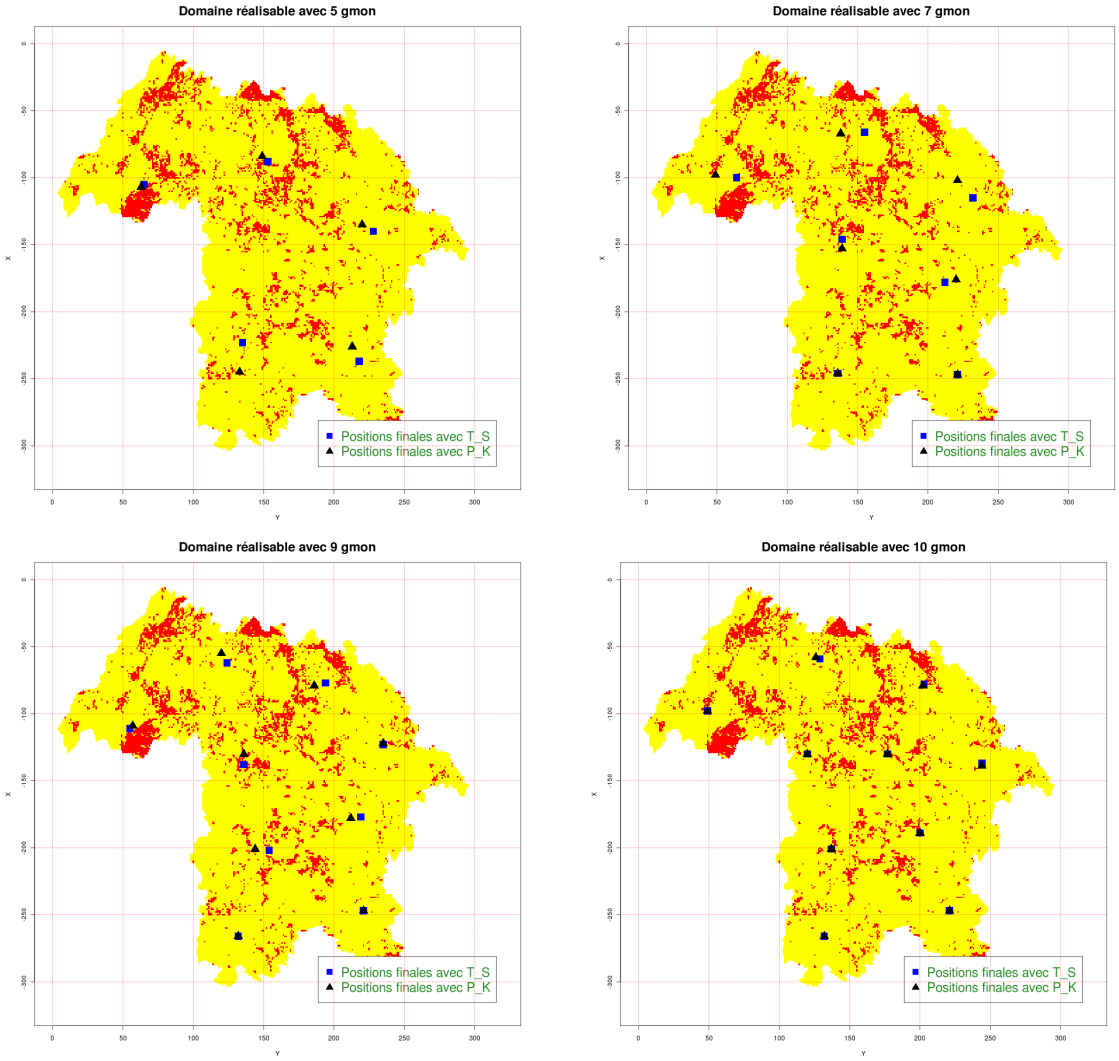

FIGURE 6.2 Comparaisons des configurations finales de balises GMON obtenues par les algorithmes T_S et P_K sur la boîte noire de l'IREQ sur la carte STM, après 1000 itérations, pour un nombre de GMON égal à 5, 7, 9 et 10.

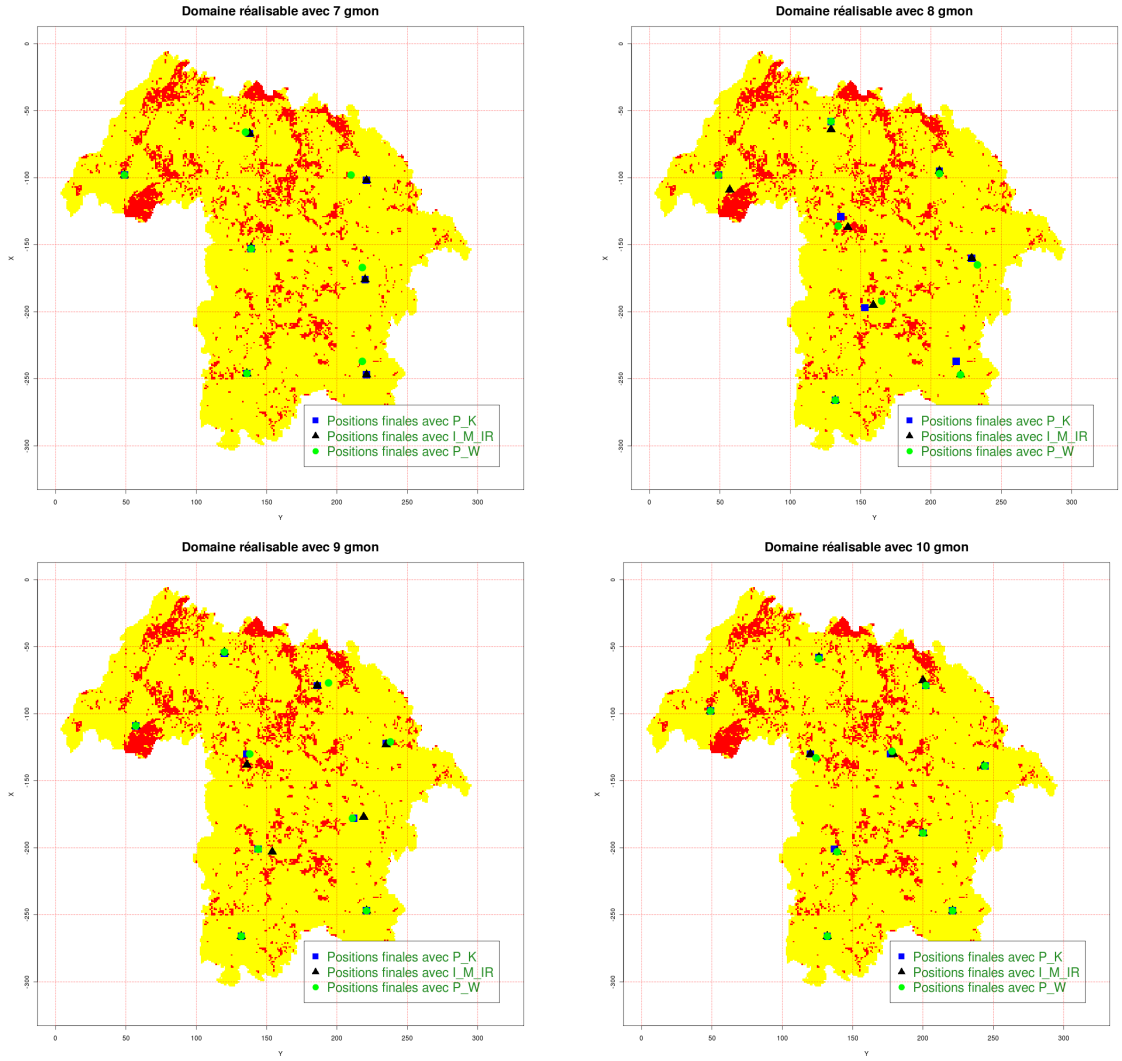

FIGURE 6.3 Comparaisons des configurations finales de balises GMON obtenues par les trois meilleures stratégies (P_K, I_M_IR et P_W) sur la boîte noire de l'IREQ sur la carte STM, après 1000 itérations, pour un nombre de GMON variant de 7 à 10.

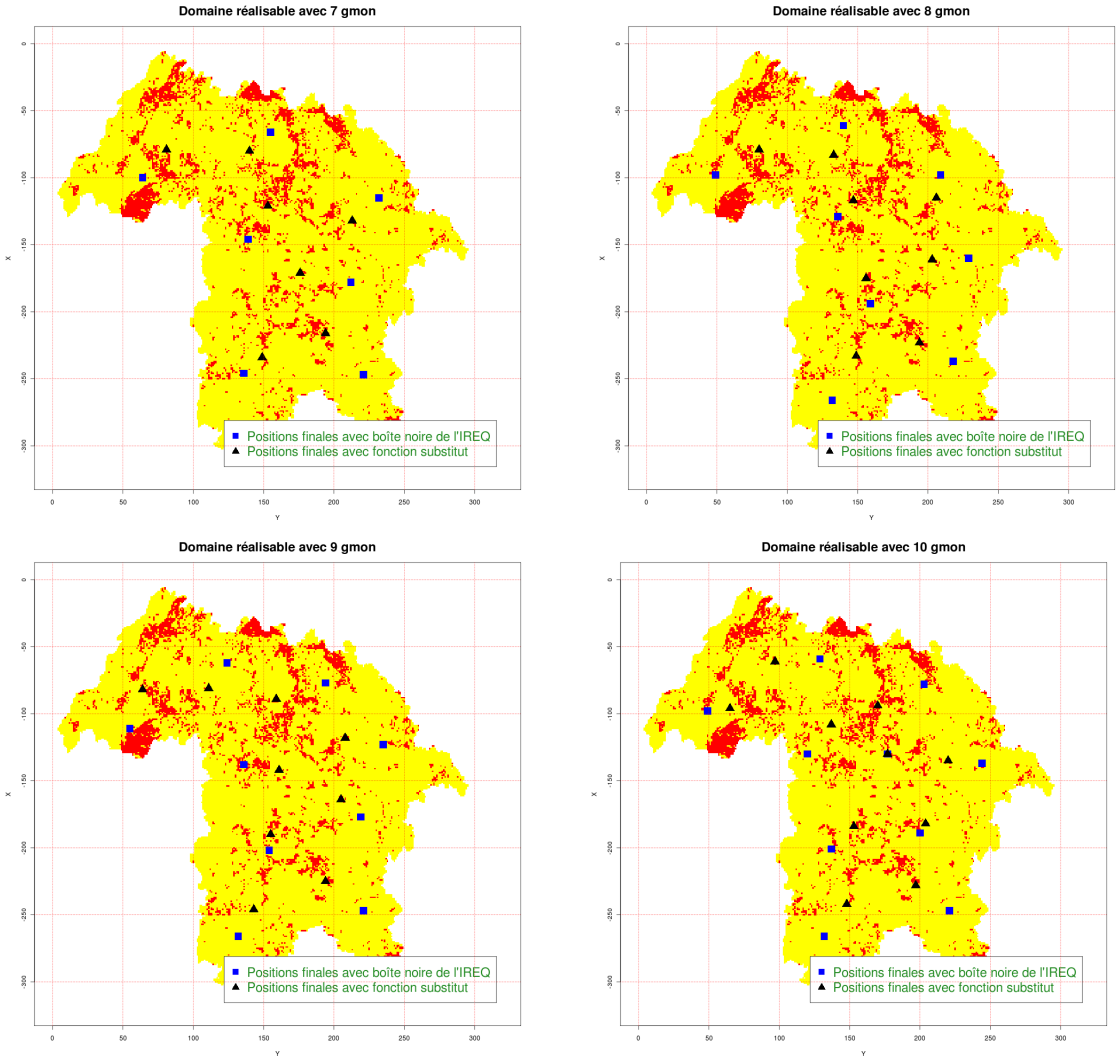

FIGURE 6.4 Comparaisons des configurations finales de balises GMON obtenues par l'algorithme T_S avec le modèle substitut et la boîte noire de l'IREQ sur la carte STM, après 1000 itérations, pour un nombre de GMON variant de 7 à 10.

Chapitre 7

CONCLUSION

Indéniablement, l'aptitude à optimiser des structures inconnues est extrêmement précieuse et constitue une préoccupation majeure de la communauté industrielle. L'optimisation de boîte noire, qui trouve son support théorique dans l'analyse non lisse de Clarke, constitue donc un cadre utile et riche, dont les limites sont sans cesse repoussées par les évolutions technologiques. Ce travail, mené sur une classe particulière d'algorithmes de recherche directe, avait pour objectif initial de prouver ou réfuter l'utilité d'une stratégie de gestion des variables particulière à la fois simple et fertile, que nous avons nommée "groupement de variables". À mesure que nous avançions, ils nous a paru souhaitable et même nécessaire de lier étroitement cette recherche à l'application qui l'avait motivée, à savoir la localisation de balises GMON pour l'estimation de l'équivalent eau-neige au Québec.

7.1 Synthèse des travaux

Nous avons étudié une extension de l'algorithme MADS : le groupement de variables, qui consiste en la gestion des sous-ensembles de variables modifiés (individuellement) à chaque itération. L'étude théorique des groupes de variables nous a permis d'exhiber deux architectures génériques d'algorithmes MADS conservant les propriétés de convergence hiérarchique de l'algorithme MADS classique : l'une est basée sur l'ajout en parallèle d'un algorithme MADS classique et l'autre sur la fusion progressive des groupes de variables.

Afin de prouver l'existence d'un potentiel résidant dans le groupement de variables, nous avons par la suite développé une centaine de variantes d'algorithmes de reconfiguration destinés à résoudre un problème de localisation de balises issu de l'IREQ, basées sur trois logiques distinctes : la *distance* entre les groupes de variables, le *mouvement* des groupes de variables, et l'*analyse de sensibilité* des groupes de variables.

Les deux premiers concepts se prêtent aisément aux cas des variables de positionnement : il est en effet possible d'adapter les algorithmes qui ont été décrits dans ce mémoire à n'importe quel problème de localisation. Le troisième concept, quant à lui, est plus général encore. Bien que nous n'en ayons étudié qu'une infime fraction (la régression linéaire), il s'est révélé performant, alors même qu'il n'exploitait pas directement la structure du problème.

L'accès à la boîte noire de l'IREQ étant fortement limité, nous avons conçu un modèle substitut pour effectuer un grand nombre de tests préliminaires qui nous ont permis de trier et sélectionner un sous-ensemble d'algorithmes *a priori* efficaces.

Nous avons finalement appliqué ces algorithmes au problème original de positionnement de l'IREQ. Disposant d'un budget en temps restreint et d'une boîte noire en cours de conception, nous n'avons pu explorer que quelques unes des possibilités étudiées. Cependant, sur les six algorithmes dynamiques testés, cinq ont permis l'obtention d'une solution de meilleure qualité qu'avec un algorithme MADS classique et tous ont été meilleurs que la Recherche par Coordonnées. Le groupement de variables a donc fourni des résultats encourageants.

7.2 Limitations de la solution proposée

Ces résultats pratiques sont à nuancer pour plusieurs raisons. Premièrement, si le cadre algorithmique proposé est général, ces résultats ne sont pertinents que pour la gamme de problèmes étudiés, à savoir les problèmes de localisation en deux dimensions avec domaine fragmenté. Ensuite, du fait de limitations propres à la boîte noire originale, le nombre de tests réalisés deumeure faible, risquant de rendre les résultats peu significatifs.

Soulignons également que les paramètres des algorithmes dynamiques présentés n'ont pas été particulièrement optimisés pour obtenir des performances. En effet, nous avons essayé de fournir un éventail d'algorithmes convergeant vers une fusion complète des variables en un nombre de séquences fixé arbitrairement (égal au nombre de balises GMON à positionner). L'obtention de résultats favorables n'en est que plus satisfaisante.

7.3 Améliorations futures

Nous souhaitons continuer, dans des travaux à venir, à mener des tests en collaboration avec l'IREQ, de façon à confirmer l'efficacité des méthodes que nous avons développées. Nous espérons d'ailleurs que l'IREQ pourra concevoir une boîte noire libérée des contraintes inhérentes au logiciel ISATIS, éventuellement accessible à l'ensemble de la communauté mathématique.

Le vaste terrain des choix algorithmiques comporte un nombre exponentiel de possibilités, nous contraignant à restreindre nos travaux à quelques cas spécifiques. Toutefois, il serait intéressant, puisqu'il est possible d'optimiser les paramètres algorithmiques grâce à l'algorithme MADS classique (cf. Audet et Orban (2006)), de tenter d'optimiser, dans de futurs travaux, les différents critères que nous avons considérés fixes, tels que le critère d'arrêt pour reconfiguration, le nombre d'exécutions menant à la fusion complète des groupes de variables, la mise à jour du treillis ou encore le choix du point de départ entre deux séquences de reconfiguration.

Il serait également profitable de développer davantage le cadre de l'analyse de sensibilité (régression, statistiques descriptives, classification, décomposition de variances, analyse en composantes principales, réseau de neuronnes) ainsi que les modèles de fonctions de score destinés à établir une relation d'ordre entre les groupes de variables. Nous nous sommes limités ici au cas sans mémoire (seule la dernière phase d'exécution est utilisée pour effectuer les statistiques), avec quelques fonctions de score (basées sur la régression linéaire).

Nous aspirons finalement à étendre notre étude, dans des travaux ultérieurs, à des problèmes sensiblement différents, tels que ceux avec variables prépondérantes ou les problèmes de localisation sur des domaines (relativement) continus.

Références

ABRAMSON, M. A., AUDET, C., CHRISSIS, J. et WALSTON, J. (2009a). Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3, 35–47.

ABRAMSON, M. A., AUDET, C., COUTURE, G., DENNIS, J. E., JR. et LE DIGABEL, S. (2003). The NOMAD project. Software available at `http://www.gerad.ca/nomad`.

ABRAMSON, M. A., AUDET, C., DENNIS, J. E., JR. et LE DIGABEL, S. (2009b). OrthoMADS : A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization*, 20, 948–966.

ALARIE, S. et GARNIER, V. (2009). Station location problem for snow-water equivalent measurement. Presentation at Optimization Days. Conference at the 2009 Optimization Days, Montreal, May 10-12th.

ALBERTO, P., NOGUEIRA, F., ROCHA, U. et VICENTE, L. N. (2004). Pattern search methods for user-provided points : application to molecular geometry problems. *SIAM Journal on Optimization*, 14, 1216–1236.

ALOISE, D., DESHPANDE, A., HANSEN, P. et POPAT, P. (2009). Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75, 245–248.

AUDET, C., BÉCHARD, V. et LE DIGABEL, S. (2008a). Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41, 299–318.

AUDET, C., COUTURE, G., DENNIS, J. E., JR., ABRAMSON, M. A., GONZALEZ, F., V.TITOV et SPILLANE, M. (2005). Optimal placement of tsunami warning buoys using mesh adaptive direct searches. Presentation at Optimization Days. `http://www.gerad.ca/Charles.Audet/Slides/Tsunami_JOPT.pdf`.

AUDET, C. et DENNIS, J. E., JR. (2003). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13, 889–903.

AUDET, C. et DENNIS, J. E., JR. (2004). A pattern Search Filter Method for Nonlinear Programming without Derivatives. *SIAM Journal on Optimization*, 14, 980–1010.

AUDET, C. et DENNIS, J. E., JR. (2006a). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17, 188–217.

AUDET, C. et DENNIS, J. E., JR. (2006b). Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17, 188–217.

AUDET, C. et DENNIS, J. E., JR. (2009). A Progressive Barrier for Derivative-Free Nonlinear Programming. *SIAM Journal on Optimization*, 20, 445–472.

AUDET, C., DENNIS, J. E., JR. et LE DIGABEL, S. (2008b). Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization*, 19, 1150–1170.

AUDET, C., DENNIS, J. E., JR. et LE DIGABEL, S. (2010). Globalization strategies for mesh adaptive direct search. *Computational Optimization and Applications*, 46, 193–215.

AUDET, C. et LE DIGABEL, S. (2009). The mesh adaptive search algorithms for periodic variables. Rapport technique G-2009-23, Les Cahiers du GERAD.

AUDET, C., LE DIGABEL, S., LEQUY, Q., SYLLA, M., ALARIE, S. et MARCOTTE, O. (2008c). Localisation de stations de mesure automatisée du couvert nival. *Proceedings of the Second Montreal Industrial Problem Solving Workshop, CRM Research report (CRM-3277)*, 9–17.

AUDET, C. et ORBAN, D. (2006). Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal on Optimization*, 17, 642–664.

AUDET, C., SAVARD, G. et ZGHAL, W. (2008d). Multiobjective Optimization Through a Series of Single-Objective Formulations. *SIAM Journal on Optimization*, 19, 188–210.

BOOKER, A. J., DENNIS, J. E., JR., FRANK, P. D., SERAFINI, D. B. et TORCZON, V. (1998). Optimization using surrogate objectives on a helicopter test example. J. Borggaard, J. Burns, E. Cliff et S. Schreck, éditeurs, *Optimal Design and Control*. Birkhäuser, Cambridge, Massachusetts, Progress in Systems and Control Theory, 49–58.

BOOKER, A. J., DENNIS, J. E., JR., FRANK, P. D., SERAFINI, D. B., TORCZON, V. et TROSSET, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17, 1–13.

BOX, G. E. P. (1957). Evolutionary operation : A method for increasing industrial productivity. *Applied Statistics*, 6, 81–101.

CLARKE, F. H. (1983). *Optimization and Nonsmooth Analysis*. Wiley, New York. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

CONN, A. R., SCHEINBERG, K. et TOINT, P. L. (1997a). On the convergence of derivative-free methods for unconstrained optimization. M. D. Buhmann et A. Iserles, éditeurs, *Approximation Theory and Optimization, Tributes to M. J. D. Powell*, Cambridge University Press, Cambridge. 83–108.

CONN, A. R., SCHEINBERG, K. et TOINT, P. L. (1997b). Recent progress in unconstrained nonlinear optimization without derivatives. 79, 397–414.

CONN, A. R., SCHEINBERG, K. et TOINT, P. L. (1998). A derivative free optimization algorithm in practice. *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September 2-4*.

CONN, A. R., SCHEINBERG, K. et VICENTE, L. N. (2009). *Introduction to Derivative-Free Optimization*. MPS/SIAM Book Series on Optimization. SIAM, Philadelphia.

CUSTÓDIO, A. L., DENNIS, J. E., JR. et VICENTE, L. N. (2008). Using simplex gradients of nonsmooth functions in direct search methods. *IMA Journal of Numerical Analysis*, 28, 770–784.

CUSTÓDIO, A. L. et VICENTE, L. N. (2007). Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18, 537–555.

DENNIS, J. E., JR. et WOODS, D. J. (1987). Optimization on microcomputers : The Nelder-Mead simplex algorithm. A. Wouk, éditeur, *New Computing Environments : Microcomputers in Large-Scale Computing*, SIAM, Philadelphia. 116–122.

DOLAN, E. et MORÉ, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–213.

ENVIRONNEMENT CANADA (2009). Le coût des inondations au Québec. `http://www.ec.gc.ca/eau-water/default.asp?lang=Fr&n=02A71110-1`. Mis en ligne le 29 juin 2009, consulté le 25 septembre 2009.

FINKEL, D. et KELLEY, C. (2004). Convergence analysis of the direct algorithm. Rapport technique CRSC-TR04-28, Center for Research in Scientific Computation.

FLETCHER, R., LEYFFER, S. et TOINT, P. L. (1998). On the global convergence of an SLP–filter algorithm. Rapport technique NA/183, Dundee University, Department of Mathematics.

FLETCHER, R., LEYFFER, S. et TOINT, P. L. (2006). A brief history of filter methods. *SIAM SIAG/OPT Views-and-News*, 18, 2–12.

FOWLER, K., REESE, J., KEES, C., DENNIS, J. E., JR., KELLEY, C., MILLER, C., AUDET, C., BOOKER, A., COUTURE, G., DARWIN, R., FARTHING, M., FINKEL, D., GABLONSKY, J., GRAY, G. et KOLDA, T. (2008). Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31, 743–757.

HALTON, J. H. (1960). On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals. *Numerische Mathematik*, 2, 84–90.

HARTIGAN, J. A. et WONG, M. A. (1979). A k-means clustering algorithm. *Applied Statistics*, 28, 100–108.

HEDAR, A.-R. et FUKUSHIMA, M. (2004). Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. 19, 291–308.

HOOKE, R. et JEEVES, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8, 212–229.

HOUGH, P. D., KOLDA, T. G. et TORCZON, V. (2001). Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23, 134–156.

IMAN, R. L. et CONOVER, W. J. (1979). The use of the rank transformation in regression. *Technometrics*, 21, 499–509.

JACQUES, J. (2005). *Contributions à l'analyse de sensibilité et à l'analyse discriminante généralisée*. Thèse de doctorat, Université Joseph Fourier - Grenoble 1.

JONES, D., PERTTUNEN, C. et STUCKMAN, B. (1993). Lipschitzian optimization without the Lipschitz constant. 79, 157–181.

KELLEY, C. T. (1999). Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10, 43–55.

LAGARIAS, J. C., REEDS, J. A., WRIGHT, M. H. et WRIGHT, P. E. (1998). Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9, 112–147.

LE DIGABEL, S. (2009). NOMAD : Nonlinear optimization with the MADS algorithm. Rapport technique G-2009-39, Les cahiers du GERAD.

LEWIS, R. M. et TORCZON, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9, 1082–1099.

LEWIS, R. M. et TORCZON, V. (2000). Pattern Search Methods for Linearly Constrained Minimization. *SIAM Journal on Optimization*, 10, 917–941.

LUERSEN, M. (2004). *GBNM : Un Algorithme d'Optimisation par Recherche Directe - Application à la Conception de Monopalmes de Nage*. Thèse de doctorat, Institut National des Sciences Appliquées, SPMI, Rouen, France.

MCKINNON, K. I. M. (1998). Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9, 148–158.

MINISTÈRE DES RESSOURCES NATURELLES ET DE LA FAUNE (2004). L'hydroélectricité. `http://www.mrn.gouv.qc.ca/energie/energie/energie-sources-hydroelectricite.jsp`. Mis en ligne en 2004, consulté le 23 septembre 2009.

NAZARETH, L. et TSENG, P. (2002). Gilding the Lily : A variant of the Nelder-Mead algorithm based on golden-section search. 22, 133–144.

NELDER, J. A. et MEAD, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7, 308–313.

POLAK, E. et WETTER, M. (2006). Precision control for generalized pattern search algorithms with adaptive precision function evaluations. *SIAM Journal on Optimization*, 16, 650–669.

POWELL, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7, 155–162.

PRICE, C. J., COOPE, I. D. et BYATT, D. (2002). A convergent variant of the Nelder-Mead algorithm. 113, 5–19.

PRICE, C. J. et TOINT, P. L. (2004). Exploiting problem structure in pattern-search methods for unconstrained optimization. 21, 479–491.

R DEVELOPMENT CORE TEAM (2010). *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

ROCKAFELLAR, R. T. (1980). Generalized directional derivatives and subgradients of nonconvex functions. *Canadian Journal of Mathematics*, 32, 257–280.

SALTELLI, A., CHAN, K. et SCOTT, E. M. (2000). *Sensitivity Analysis*. Wiley.

SALTELLI, A. et SCOTT, E. M. (1997). Guest editorial : The role of sensitivity analysis in the corroboration of models and its link to model structural and parametric uncertainty. *Reliability Engineering and System Safety*.

SALTELLI, A., TARANTOLA, S., F.CAMPOLONGO et RATTO, M. (2004). *Sensitivity analysis in practise*. Wiley.

SAPORTA, G. (2006). *Probabilités, analyse des données et statistique*. Technip, Paris, seconde édition.

SOBOL, I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments*, 1, 407–414.

SPENDLEY, W., HEXT, G. R. et HIMSWORTH, F. R. (1962). Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4, 441–461.

SRIVER, T. A., CHRISSIS, J. W. et ABRAMSON, M. A. (2004). Pattern search ranking and selection algorithms for mixed variable stochastic optimization. Preprint.

STEPHENS, C. P. et BARITOMPA, W. (1998). Global optimization requires global information. *Journal of Optimization Theory and Applications*, 96, 575–588.

TAPSOBA, D., FORTIN, V., ANCTIL, F. et HACHÉ, M. (2005). Apport de la technique du krigeage avec dérive externe pour une cartographie raisonnée de l'équivalent en eau de la neige : Application aux bassins de la rivière Gatineau. *Canadian journal of civil engineering*, 32, 289–297.

TORCZON, V. (1997). On the Convergence of Pattern Search Algorithms. *SIAM Journal on Optimization*, 7, 1–25.